June 1988 Issue 20
U.S.A. $3.50
Canada $4.75

ST LOG

THE ATARI ST MONTHLY MAGAZINE

New Product Reviews

BASIC Draw

Mouse-

urce

ST Font Printer

And Keep Your Cool.

Introducing the ST Hard Drive System from ICD that refreshes your memory better than any other ST hard drive around. *No problem.*

It's the drive that not only looks cool, but stays cool too. All because of a built-in fan that knows exactly how to beat the heat and maintain a calm, cool and collected environment. . . even in *your* most heated situations. *No sweat.*

And, it's the hard drive that takes a refreshing approach to aesthetic case design as well. See for yourself. It's easy on the space, fitting perfectly under the monitor. And it's easy on the eyes, tailored to look great in the company of your Atari ST. With adjustable front legs, your monitor gets the lift it needs for comfortable viewing. *No strain.*

Despite a sleek and compact exterior, the ICD ST Hard Drive System is packed full of overwhelming enhancements. Like an internal clock that tags each file with up-to-the-minute time and date information. Not to mention expansion capabilities that welcome the connection of up to six SCSI devices and daisy-chaining Atari's DMA Bus (ACSI). It's available in more memory capacities than you can imagine. With storage ranging from 20 megabyte systems up to 2801 megabytes. And, there's dual drives too, that double your protection and double your confidence. *No stress.*

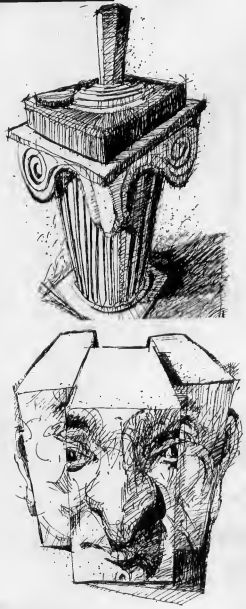
So, the next time you think about a hard drive for your Atari ST, think about the countless ways we can refresh your memory. It's the only drive worth remembering. Because it's from ICD. *No wonder.*

For further product information, please call or write for our catalog today.

1220 Rock Street
Rockford, IL 61101-1437
(815)968-2228
MODEM: (815)968-2229
FAX: (815)968-6888

ST LOG
THE ATARI ST MONTHLY MAGAZINE

Front cover model is not an actual U.S. military officer. Representation for illustration purposes only. Cover photography: Ladi Von Jansky. Covermodel: Norman Funk. Screen illustrations: Cinemaware/SDI for the Atari ST.



FEATURES

REVIEWS

COLUMNS

ST-LOG (ISSN 0890-9601) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. U.S. subscription $28 for one year (12 issues), $52 for two years (24 issues, $76 for three years (36 issues). Foreign subscriptions: add $7 per subscription per year. Single copy $3.50 (add $1 for postage). Change of address: six weeks advance notice, and both old and new addresses are needed. POSTMASTER, send change of address to: **ST-LOG** Magazine, P.O. Box 16928, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

EDITORIAL

by Clayton Walnum

Man, it's really starting to bug me. Everywhere I go, it's gloom and doom. It's never ceased to amaze me how the human race has to be miserable to be happy. The hottest news items are usually tales of violence, while the good things in life go largely overlooked.

Unfortunately, things aren't much different in Atariland. It's become impossible to discuss computing on our machines without getting inundated with complaints and dire predictions. Every message base on every on-line service is filled with dark mutterings, every Atari discussion ends up as a none-too-supportive gripe session.

There's something we need to consider: the self-fulfilling prophecy.

A self-fulfilling prophecy is a phenomenon that occurs when someone believes something so intently that he unconsciously causes it to happen, either by giving up and doing nothing to prevent it or by actively bringing it about. And the self-fulfilling prophecy is not a phenomenon that affects only individuals; large groups can just as easily fall under its spell—take your average *group* of computer users, for instance.

What's the point? Word-of-mouth advertising and the self-fulfilling prophecy share the same bed. That's one of the great dangers inherent in this form of advertising. And when a company—rightly or wrongly—depends to a great extent on word-of-mouth to sell its product, it's taking a major gamble.

Atari seems to have taken that gamble, and I'm not sure it was wise.

You see, we Atari users have always been the underdogs; there's nothing new in that. When everyone else was buying Commodores and Apples, we stuck by our 8-bit machines with the undying loyalty of lionesses protecting their young. We didn't let the rest of the world interfere with our pleasure.

As ST owners, we seem to have metamorphosed into an entirely different creature.

I think Atari has been counting on that loyalty to sell its STs.

But as ST owners we seem to have metamorphosed into an entirely different creature. We can't accept that Atari isn't IBM or Apple. We refuse to be happy with what we have and instead spend most of our time wishing our STs were something they're not. We have expectations for Atari that can never be fulfilled and, like a parent that wants great things for its child—regardless of the cost or the child's desires—we place ourselves in the position of watching the whole thing blow up in our face. The more Atari drifts from the image we want for them, the more restless we get; the more restless we get, the more we grumble; the more we grumble, the more we assure the self-fulfilling prophecy.

No, I don't agree with everything Atari's done. But I am willing to assume that they, having a greater knowledge of their company's resources, know what's best for them and their business.

At any rate, Atari's not going to change its image overnight, and it's unreasonable for us to expect them to; just as it's unreasonable for us to ask them to ignore profit opportunities (for instance, video games or the European computer market) in order to become what we want them to be. Atari is going to run their company according to their own rules, like it or not. We can only hope that those rules have been considered carefully.

Fashions come and fashions go. Today it's fashionable to criticize Atari. This bad press succeeds in only one thing: creating the self-fulfilling prophecy. Keep it up. The guy next door is listening very carefully, and he'll buy a Mac or an Amiga. That'll not only be his loss, but ours as well.

ST-Log Staff

Publisher: Lee H. Pappas. *Executive Editor:* Clayton Walnum. *Art Director:* Eddy Herch. *Managing Editor:* Dean Brierly. *East Coast Editor:* Arthur Leyenberger. *Midwest Editor:* Matthew J.W. Ratcliff. *West Coast Editor:* Charles F. Johnson. *Contributing Editors:* Michael Banks, Ian Chadwick, Andy Eddy, Arnie Katz, Bill Kunkel, Maurice Molyneaux, Steve Panak, Douglas Weir, Joyce Worley, Copy Chief: Katrina Veit. *Copy Editors:* Anne Denbok, Sara Bellum, Pat Romero. *Typographers:* Judy Villanueva, David Buchanan, Klarissa Curtis. *Contributors:* Tom Castle, Frank Cohen, Betty D. DeMunn, Michael Donahue, Colin Faller, Kevin Kennedy, E.H. Wysocki. *Production Director:* Donna Hahner. *Production Assistant:* Steve Hopkins. *Advertising Production Director:* Janice Rosenblum. *Subscriptions Director:* Irene Gradstein. *Vice-President, Sales:* James Gustafson.

U.S. newsstand distribution by Eastern News Distributors, Inc., 1130 Cleveland Rd., Sandusky, OH 44870.

Where to write

Correspondence, letters and press releases should be sent to: Editor, **ST-LOG**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ST-LOG** P.O. Box 16928, North Hollywood, CA 91615. Or Call (818) 760-8983

Correspondence concerning a regular column should be sent to our editorial address, with the name of the column included in the address. We cannot reply to all letters in these pages, so if you would like an answer, please enclose a self-addressed, stamped envelope.

Advertising Sales

J.E. Publishers Representatives — Los Angeles: (213) 467-2266.
San Francisco: (415) 864-3252. Chicago: (312) 445-2489.
Denver: (303) 595-4331.
6855 Santa Monica Blvd., Suite 200, Los Angeles, CA 90038.
New York: (212) 724-7767.

Address all advertising materials to: Janice Rosenblum — Advertising Production, **ST-LOG**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

Permissions

Subscriptions

ST-LOG, P.O. Box 16928, North Hollywood, CA 91615; or call (818) 760-8983. Payable in U.S. funds only. U.S.: $28.00-1 year; $52.00-2 years; $76.00-3 years. Foreign: add $7.00 per year per subscription. For disk subscriptions, see the cards at the back of this issue.

Authors

When submitting articles and programs, both program listings and text should be provided in printed *and* magnetic form, if possible. Typed or printed text copy is mandatory and should be in upper- and lowercase, with double spacing. If a submission is to be returned, please send a self-addressed, stamped envelope to **ST-LOG**, P.O. Box 1413-MO, Manchester, CT 06040-1413.

REVIEW

RoadWar 2000

SSI
1046 North Rengstorff Ave.
Mountain View, CA 94043
512K Disk $39.95

by Steve Panak

The underlying theme of this new game from SSI is somewhat less than original. Lately we've been inundated with post-nuclear disaster worlds filled with rebel gangs and mutants battling on the highways and byways of tomorrow. I've seen it in a number of books and movies, most successfully in the *Mad Max* series. Because of this large customer base, any game trying to cash in on this guaranteed market is going to have to be really good. Fortunately, **Roadwar 2000** goes that extra mile.

The time is the year 2000. The place, North America. Your goal, to control the continent and save the world. Your obstacles are numerous foot and road gangs, including cannibals, renegade national guardsmen, and mutants. A biological, rather than nuclear, attack has turned the clocks back hundreds of years and made our land one in which only the strong survive. Of course, later, when we're weakened, is when the bombs start dropping, poisoning the cities and ecosystems, and generally lowering everyone's standard of living.

Roadwar 2000 starts with this popular premise and adds ease of control to create a game that is both intriguing and enjoyable to play. After booting the disk your screen display fills with a map of a region of North America. The right portion of the monitor contains a status area showing your group's vital statistics, such as the amount of supplies and fuel your band possesses. At the top are a number of pull-down menus from which you issue all of your commands. If you so desire, you can also control the game with the keyboard. Throughout most of the game, you use only four options. By clicking the mouse in the direction you wish to move, your army advances down the road (or, sometimes, as circumstances require, off the road).

After each move (which will cost you time, fuel, money, and possibly the lives of your entire band), you usually will want to do any or all of the following: search for loot, search for people, and search for vehicles. You require loot to keep moving (gas and tires) and to keep alive (food, medicine and weapons). You require new and stronger vehicles and people to replenish those lost in combat.

The main object of the game is to bring together eight agents who will hopefully be able to produce an antidote to the disease that has infected the populace. You do this by assembling a group containing a number of fighters and vehicles, as well as a doctor, a drill sergeant, and a politician, who increase party health, morale, and charisma, respectively. As you build your army, you move between North America's major cities, looting, recruiting and controlling these metropolitan areas. Only after 50+ hours of conflict will you complete this game. Maybe. But if searching and looting were your only tasks, this game would quickly become a bore. Fortunately, (I guess) you are not the only ones on the roads.

From time to time as you travel down the highways, you'll run into rival road gangs. When this occurs, there is no negotiation, no surrender. You must fight, and only one gang survives. As each confrontation occurs, you choose one of two combat options, tactical or abstract. An abstract battle simply computes and displays the winner. But if you choose tactical mode, you control all aspects of the skirmish. You disperse your men among the vehicles, arrange the vehicles into battle formation, and then engage the enemy. In tactical mode, you can choose a quick resolution, which makes all the combat strategy decisions for you, or you can completely control the fight, training fire on, and ramming and boarding enemy vehicles of your choice. Upon the successful completion of full tactical combat, the number of vehicles allowed in your gang is increased by one, up to a maximum of 15.

In addition to searching, moving, and fighting, other menus contain even more play options. Of course, you can save and restore games in progress. What is really great is that the game disk is unprotected (let's not pirate it!), and on your back-up copy you can save your position. This makes saving quick and effortless, with no need to ever swap disks. As far as play options go, you can examine the strength and size of your gang, get up-to-date statistics on supplies, transfer supplies in and out of storage caches (which may be established in cities), fix tires, and abandon vehicles and supplies.

Being a simulation, this game allows for a high degree of realism. You have the choice of 19 vehicles, from motorcycles to tractor-trailer rigs. Each vehicle is rated for speed, maneuverability, armor, as well as crew, supply, and fuel capacity. Similarly, each member of your gang (which could number in the thousands) is rated and promoted through five ranks of increasing skill. Injured people and damaged vehicles may be repaired, and throughout the land there are certain special items, such as snow tires, which help you along your way.

If all this sounds like a lot, rest assured it is — and that's why the game is so good. The ease of play keeps you in high gear. You quickly learn the operation of the pulldown menus, and choices are made effortlessly with the mouse.

Unfortunately, learning what to do is another thing. The manual, while superbly written, concentrates more on telling a story than explaining how to play the game. Even after reading it cover to cover you still may not know just what you're supposed to do. I would have liked to have seen some sort of quick-start chapter, defining the goal and providing useful hints.

Due to the long nights I spent trying to conquer this game, I have to give a thumbs up to Roadwar 2000. Although its goal was not immediately apparent, it was easy to play, and, best of all, easy to enjoy. It is just another in a long line of games for the ST which establish the machine as the premier gaming computer available today. Jump behind the wheel of Roadwar 2000—you'll love the ride.

REVIEW

Monitor Master

Practical Solutions
1930 East Grant Road
***Tucson*, AZ 85719**
(602) 884-9612
$49.95

by Maurice Molyneaux

Don't you just *hate* switching cables? If you have both an RGB and a monochrome monitor, or if you have a monochrome monitor and use a television for color programs, it can be a real pain to have to constantly pull out and plug in monitor cables. I did this for a year and a half, then one day I finally broke down and did something about it. I bought **Monitor Master,** and my cable switching days were over.

Monitor Master (MM) is a video switchbox, colored Atari gray, and just a bit bigger than your ST's mouse. Its only control is a single big, black button on its face. Around back you'll find a cable that plugs into your ST's monitor port and four other jacks. Two jacks are designed to take ST monitor cables and are labeled "color" and "mono." In addition to these are two standard RCA phono jacks, labeled "audio" and "video." If you hook a standard stereo cable into the audio jack, you can then pipe sound from your ST to another monitor, or through your stereo system (**Starglider** takes on a whole new dimension when piped through some *good* speakers). If you have an RF-equipped ST (most 520STs and the rare 1040STfm) you can hook a composite monitor, VCR, etc., into the video jack. If you use a monochrome monitor and a TV, you just plug the monitor into the proper jack and hook up your TV as always.

The black button (MM's only control) is used to toggle between color and monochrome. Normally, if you're just using a TV by means of the RF modulator (if your machine has one) rather than an RGB monitor, you must still disconnect your monochrome monitor to run programs in low or medium resolution (though medium looks terrible on most TVs and composite monitors). When MM's button is pushed in, the monochrome monitor is disabled, and the RGB port, composite video jack or RF modulator can be used for output. When you want to go to monochrome, pushing the button so it pops into its out position disables color modes.

For those of you who didn't know, the ST's monitor jack has a pin called "monochrome detect" which, when connected to a monochrome monitor, lets the ST know it's in monochrome mode. Since the monochrome display signal is different than that for color modes (for example, the mono monitor has a screen refresh rate of 70 times per second as opposed to the color's 60), the ST will reset itself whenever a change occurs in the status of the monochrome detect pin. So, if you press MM's button while your computer is running, it will reboot. However, while doing this will reset the computer, I recommend that you turn off your ST whenever you perform this toggle, as doing so with the power on causes needless wear and tear on your computer.

There's not much else to say about **Monitor Master**. It does its intended job and does it quite well. The composite audio/video output is especially nice, because it allows you to hook up a VCR and tape your color graphics (only low resolution is worth the effort), in addition to hooking up to stereos and composite monitors. If your ST lacks an RF modulator (you can tell by looking at the back of the machine; if there's a jack with a TV icon above it, and a small slot with a channel 2/3 selector switch, then you've got it—if not, you don't), then you clearly can't use the composite video output.

However, Practical Solutions will soon be releasing the **Video Key**, a device which converts any ST's RGB output to a high-quality composite video signal (so don't you 1040ST and Mega ST owners despair). They also currently offer six-foot microfloppy disk-drive cables as well as numerous video cables; including special cables to hook up non-Atari RGB and multisynch monitors. To put icing on the switchbox cake, if you're tired of switching cables between your mouse and a joystick, Practical Solutions is coming out with **Mouse Master**—the function of which I daresay we can all guess!

So, if you're tired of those monitor cable-switching blues, get yourself a **Monitor Master**.

READER COMMENT

I just bought an Atari 1040ST, and I love it! It's going to take me a while to get used to using it (this is my first computer), but I'm looking forward to becoming an "expert" soon. I'm having a little difficulty with the built-in disk drive, though. Sometimes when I boot the computer, the disk drive won't stop turning. The busy light stays on, and I'm afraid that if I try to remove the disk and put in another one, I'll damage them. Do other people have this problem? Is there anything I can do?—Mary Williamson
Dickson, TN

The problem you're having is not untypical of the 1040 ST. The fix is simple. When the drive keeps spinning just pop the disk out and reinsert it. This will usually cause the drive to go back to normal operation. In your case, you don't have to worry about doing damage to your disk, because nothing is happening inside the drive except the motor spinning. However, you should never remove a disk from the drive when it's being written to (when something is being saved). This could cause the data on the disk to become scrambled.

I have a Supra 60MB hard drive to go along with my 1MB 520-ST. Every once in a while, when I turn on the computer, the hard drive icons don't come up properly. (I've got the drive partitioned into four logical drives C, D, E and F). I don't know if there's something wrong with the hard drive or with the boot software. The only way I've found to get it to work again is to turn everything off for a couple of minutes, then reboot it. Am I doing something wrong?
—Greg Albertson
Reno, NV

You're probably not doing anything wrong, Greg; in fact, you discovered the proper solution to your problem on your own, although usually you don't need to turn off the entire system. Just turning off the hard drive for a few seconds and then rebooting will usually correct this mysterious malady.

I'm very confused (no wisecracks, please). In the editorial for the April ST-Log, Lee Pappas stated that the magazine's design was going to improve by "adding more color and incorporating more creative layouts." Flipping through the issue, I can't see any changes at all. ST-Log looks exactly the same.
—Arnold Richards
Columbia, SC

The key words in that editorial were "in the months to come." It takes a great deal of time and planning to redesign a magazine. Also, much of the work for the April and May issues of ST-Log were completed at the time of the magazines' sale to the new publisher, and we decided, in order to get the magazines on the stands as fast as possible, to use the completed material. There were no changes at all done to the April issue. By now, you've received the May issue, and I'm sure you've noticed the new cover style and logo. The June issue will be the first issue that our new artists have been able to put their full efforts to. And, believe me, when you pull it out of your mailbox, you will see the difference.

Everyone these days is talking about desktop publishing, but I'm not really sure what it is they're talking about. I know they mean the process of publishing newsletters without having to go to a professional printing company, but it seems to me that there is more to it than that. I've got a 520ST, along with the usual assortment of word processors, games, etc. But I don't have any software that mentions anything about desktop publishing. What exactly is desktop publishing? I currently do up a newsletter for a stamp collector's club. I "publish" it by typing it with ST Writer, printing out the pages on my Epson printer and then photocopying the result. Can I use my ST in a better way to help me put out this newsletter?

By the way, I want to congratulate you on a fine April issue, Mouse-ka-mania is a super program! And CHKDSK has already helped me repair dozens of my floppy disks. Keep up the great work!
—Robert Ford
Warwick, RI

You have the basic idea behind desktop publishing. In a general sense, it means using your computer to publish newsletters, forms and any other type of document without having to resort to fancy typesetting equipment. However, whether or not you utilize the services of a professional printing company has very little to do with the process of desktop publishing. In many cases, documents that were designed using desktop publishing software are printed just like any other publication—at the printers. It's how you create your "camera-ready copy" (the "master" from which your publication will be printed; in your case, the pages that you photocopy to publish your stamp collector's newsletter would be your version of camera-ready copy) that differentiates desktop publishing from the more conventional publishing methods.

A "professional" publication is designed and created using very expensive typesetting and photographic equipment. And the entire typesetting process is costly as well (especially if you need to do a lot of corrections to the "galleys," typeset copy in a preliminary form). For instance, if you were to bring a 20-page newsletter to a typesetter service, it would probably cost you nearly $1000 to get your camera-ready copy.

This cost is frequently too high for the casual publisher, so desktop publishing came into being. What you're doing with your newsletter is, technically, desktop publishing, because you're using your computer to design the document. However, what you seem to be unaware of is that there are several programs designed especially for the desktop publisher. They allow you not only to type in text, but also to incorporate graphics, columnize and "justify" your copy, design headlines, use different "fonts" (character designs) and "point sizes" (a character measurement) and utilize many other useful functions available only through quality desktop publishing software.

There are several desktop publishing packages available for the ST, including SoftLogik's Publishing Partner, Timeworks' Publishing ST, MirrorSoft's Fleet Street Publisher and Migraph's Easy-Draw (this last isn't really a desktop publishing program, but many people use it as one). All these programs (except Publisher ST, which is brand new) have been reviewed in past issues of ST-Log. You might also like to take a look at Maurice Molyneaux's article "Page Perfect" in ST-Log #17 for a mini-tutorial on desktop publishing.

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $105

SAVE TIME AND MONEY SUBSCRIBE TO ST-LOG

SAVE $14 OFF THE COVER PRICE WITH THE CONVENIENCE OF HAVING ST-LOG DELIVERED DIRECTLY TO YOUR DOOR *BEFORE* IT EVEN HITS THE NEWSSTANDS! GET THE MOST OUT OF YOUR COMPUTER

SUBSCRIBE TO ST-LOG TODAY!

☐ 1 YEAR @ $28 — SAVE $14! MCFWW
FOREIGN — ADD $7 PER YEAR
☐ 1 YEAR WITH DISK @ $105 DCFWW
FOREIGN — ADD $15 PER YEAR

☐ PAYMENT ENCLOSED ☐ BILL ME
CHARGE MY: ☐ VISA ☐ MC #_____

EXPIRATION DATE SIGNATURE

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615. Offer expires August 31, 1988. Your first issue will arrive in 6 to 8 weeks.

WATCH FOR IT!

APPLICATION

Medium or High Resolution

ST Font Printer

by Charles F. Johnson

This month ***ST-Log*** *is proud to present the second-place winner in our ST programming contest. Due to the size of the program, the source listings could not be printed in the magazine. The program (and all associated files) is available on this month's disk version and on the ANALOG Atari SIG on Delphi.*

ST Font Printer is a general purpose printing utility that will let you print any ASCII text file (up to 100K in size) to an Epson/Gemini or IBM-compatible printer, using a redefined character set or the default ST system font (the one you see on the screen in low- and medium-resolution color modes). Fonts can be printed in two sizes (single or double width). There is a Type-a-Line feature that lets you enter a line of text from the keyboard and print it with any font, very handy for titles. You can also create professional-looking four-line address labels using common 15/16 by 3½-inch tractor-feed labels, up to 99 at a time (this feature can also be used to address envelopes).

The program allows you to set some of your printer's special features to aid in producing a neatly formatted printout. You can set the left margin and the line spacing, and skip over the perforation on tractor-feed paper. Text files with embedded TABs, such as those produced by the **MicroEMACS** editor, will print with correct column alignment.

ST Font Printer supports two types of fonts; fonts created by the **DEGAS** drawing program, or any 8-bit Atari font. The program also allows you to convert fonts from one format to the other. This is a fully GEM-based program written in 68000 assembly language; it will work on any ST with TOS in ROM, in either medium-res color or monochrome modes.

How to use it

To run **ST Font Printer**, its GEM resource file (STPRINT.RSC) must be in the same directory as the program. Just double-click on STPRINT.PRG and off you go! The "ST FONT PRINTER" window will open, a title box will display, and the top menu bar will show the choices Desk (actually, an Atari logo), Exit, Fonts, Print, and Options. Click on the Continue button in the title box, and it will zoom away; now you can point at any of the menu choices causing a sub-menu to drop down. When you click on a drop-down menu item, the window's information line will show the option you've selected. The window's information line will display all the current print settings and is updated every time you change something with the Set Print Options selection. Here are explanations of the various menus and sub-menus:

Desk

ST Font Printer—Click on this selection to re-display the title box (in case you forget the name!).

Exit

Quit—Does just what it says, exits to the GEM desktop. You will see a dialog box asking you to confirm your decision. You can also click on the close box in the upper-left corner of the window to exit program.

Fonts

Load DEGAS Font

Load Atari 8-bit Font—Choose one of the font-loading functions, and an Item Selector box will appear. The default extensions for the different font types are ".FNT" for DEGAS and ".FN8" for 8-bit fonts. The window's title line shows which option you've chosen, as a reminder. To load a font you can either click once on the name of the font, and then click on the OK button, or simply double-click on the name itself; the Cancel button will abort the operation.

If you wish to change the drive or pathname, type an up-arrow to move the cursor to the directory line, and edit the drive and pathnames, but don't press RETURN when you're done. Instead, move the mouse cursor inside the file window and click once. This will show the new directory. (This is an inconsistency in Atari GEM—the IBM version of the Item Selector box will let you press RETURN after editing the directory line.) If you change the directory line, be sure to include a full, legal GEM pathname, such as "*A:\FONTS*.SET" (the backslash "\" after the colon is important). The program remembers the directories for each type of font so you'll only have to do this once. (You may also change the current drive for all file accesses by using the Set Current Drive option; see below.)

When the font is finished loading it will be displayed in a box at the bottom of the screen, and its name will be printed on the right side of the menu bar. Bear in mind that when you first run **ST Font Printer**, the custom font printing option is disabled and all printing operations will use the ST system 8 by 8 font. To print with a font you've loaded, you must go to the Set Options menu and enable it first (see below).

Write DEGAS Font

Write 8-bit Font—These options let you convert a font from 8-bit to DEGAS format, or vice versa. They are disabled (shown in lighter type) when the program is started; loading a font will enable them. When you choose to write (store on disk) either type of font, the program will present you with the famed Item Selector box once again. The current font filename is used as the default name, with the correct extension (".FNT" or ".FN8") automatically appended to it. If the font name already exists on the disk, you will be given the chance to change your mind

ST Font Printer

or continue and over-write the existing file.

Show User Font—When you first run the program this option is disabled, like the Write functions above. After you load a font file, it will be activated, and it'll display the name of the current font. Clicking on this selection will cause all subsequent drop-down menus and dialog boxes to use the current font for their text; the next time you pull down this menu, it will say Show System Font, to allow you to switch back to the default ST system font. Please note that the font name shown in the window's info line is the one that will be printed, not the one you see on-screen. To set the front to be printed see below, under Set Print Options.

Print

Print File—Choose this and you'll see another Item Selector box; this one displays all files with a ".DOC" extension. Make sure your printer is ready to go, and select the file. Another dialog box will appear to allow you to select options for a title line. You can print the filename, date, and time in a special line of inverse text at the top of your printout, or choose not to print a title line at all. Click on OK—the mouse cursor will change to the "busy bee" and your printer will start chuckling away (actually, my printer sounds more like a screaming banshee). If you want to abort the printout, just press the Undo key.

NOTE: **ST Font Printer** will display only print straight ASCII text files, with a maximum line length of 120 characters. To print an **ST Writer** document, you should use the Print option from the main ST Writer menu to print it to a disk file. Since ST Writer prints to disk with full formatting, you'll probably want to set the Skip Perforation feature to OFF. If you use **1st Word** on the other hand, save your file with Word Processor mode turned off. This creates a text file with no formatting at all, so the Skip feature comes in very handy here.

Type a Line—Lets you enter a line from the keyboard that'll be printed using the current option settings displayed in the window's information line (see below). After entering the line, you may either use the mouse and click on the OK button, or just press RETURN. This feature can be used to print titles and comments before or after your listings.

Address Labels—Choosing this option brings forth a dialog box resembling a label, with four lines in which to enter text. To move from line to line, don't press RETURN; use the arrow keys to move up and down. You can also set the number of labels to print, in a box at the lower left. Just click on the up or down arrows and the value will change accordingly. The Undo key will abort a printout, just as with the Print File option. IMPORTANT: To correctly align the labels you should set Skip Perforation to NO and set line spacing to 1/6 inch (see Set Options), and use labels that are 15/16 inch high, so it is exactly one inch from the top of one label to the next.

Options

Set Print Options—This lets you set some of the printer's special features. A dialog box will appear with several toggle-able buttons and an editable text field. You may choose to print with small or large (double width) characters, set Skip Over Perforation, choose which font you will print with, initialize the printer after printing a file, choose between several line spacings, or set the left margin wherever you like.

When everything is the way you want it, click on the OK button.

To set the left margin, click on the up or down arrows next to the box. Click and hold the mouse button on the arrows to adjust the value quickly.

If you choose not to initialize the printer, the perforation setting is retained for successive printing operations.

The Options dialog box has a Cancel button that will reset the options to their initial states without exiting the box, in case you should happen to change your mind. When everything is set the way you want it, click on the OK button. Notice that the window's info line changes to reflect any choices you've made.

Set Printer Type—This option lets you choose between two types of printers; a standard Epson or Epson-compatible printer (such as the Gemini 10-X) or an IBM-compatible printer. If you have a Star SG-10, you can enable its IBM mode by turning DIP switch 2-2 off.

Set Current Drive—Lets you select the drive which will be used for all file accesses. Up to 16 drives are supported, so you can use Ramdisks, hard disks, etc.

ST Font Printer is pretty well error-trapped. If an error occurs during a disk operation, you'll see a box containing a description of the error in English (not just something like "TOS error #33!). And if your printer isn't ready (power off, not connected, off-line, etc.) you'll see a box telling you that. If you get an "Insufficient memory" message, re-boot your system without any accessories or Ramdisks installed.

NOTE: Some fonts created with the DEGAS font editor will not look right with **ST Font Printer.** This is because DEGAS uses an 8 by 16 font matrix, and **ST Font Printer** expects its fonts to be in an 8 by 8 matrix. Thus, when it loads a DEGAS font, it discards every other byte in the matrix, creating an 8 by 8 grid from an 8 by 16 one. If some of the characters look a little strange, you can edit them with the DEGAS font editor. However, most of the DEGAS fonts I've tried do not need any editing, including the ones that are supplied with the DEGAS package.

If you use the skip-over-perforation feature, I recommend you position the printhead about two line feeds below the perforation before you start printing. This will ensure that each page of text is centered. For address labels, remember to use the settings described above. You'll have to experiment a bit to find the best starting point for label printing.

Charles F. Johnson *is a professional musician and, now, a semi-professional computer programmer/reviewer/author. He lives in Los Angeles with his wife Patty, and Spike, the world's most intelligent cat. Charles is a SYSOLP on the ANALOG PUBLISHING Atari SIG on Delphi; his user name is CFJ.*

Expandable HARD DRIVE Kits fit the MEGA too!

Install your own ST506 (PC clone) compatible drives. Two case styles with two different host adapters are available:

1. 10" x 6.5" x 15" with full SCSI interface
-room for up to 5 1/2 height 5 1/4 hard, floppy or tape drives
-150 W power supply
-controller for up to 4 hard drives (Adaptec 5500)
- mounts on floor,under desk or on desktop

Complete kits, ready to plug in 2 drives:
10 MB - $485 30MB - $675
20 MB - $595 40MB - $845
no drive, install your own $385

2. 13" wide x 15" deep x 3" high, MEGA form factor, single port host adapter:
-room for 3 1/2 height 5 1/4 or combination of full - 1/2 height - 3.5" drives
- 65 W power supply w. fan
- can be placed under the monitor

Compl. kits ready for 2 drives:
10 MB - $395 30 MB - $615
20MB - $525 40 MB - $795

No Drive...install your own$295

3. When space is at a premium...

Latest 3.5" technology, RLL with embedded SCSI contr., takes full advantage of the SCSI bus, extremely high transfer rates, almost instantaneous response, details couldn't be finalized at press time, write or phone for latest catalog!

4. Upgrade your 520STor 1040ST/520STfm to the MEGA standard, separate keyboard and CPU, gain space for up to 3 5 1/4 - 3.5" hard or floppy drives.

Kit for 520ST with all parts, central power supply, delay for operation with hard d. $345

Kit for 1040/520STfm $295

More details with our catalog!

Host Adapter cards, plug directly into the DMA port, come standard with a 6' cable, up to 20' feet and additional daisy chained connectors opt.

1 port - $79 full SCSI - $119

Software for form. + part. incl.

MEMORY UPGRADES for520, 1040/520STfm:

Up to Four Megabytes on ONE board with NO SOLDERING!!!

Expandable 520ST boards come fully socketed and use 256k DRAMS to upgrade to 1MB and 1 Megabit DIP's for 2.5 and 4 MB upgrades. Go from 1 to 2.5 or 4 MB by install. 16 or 32 1 Mb DRAM's!

1040ST/520STfm boards use only 1 Mb chips and upgrade to either 2.5 or 4 MB.

All memory boards fit under the R/F shield with the CPU completely available for any future enhancements (blitter, coprocessor etc.) and come with a ONE YEAR ltd. warranty. Installation is totally solder-free and easy with detailed illustrated instructions.

520A	socketed, no RAM	$129
520B	1 MB, socketed	$229
520C*	2.5 MB, socketed	$495
520D*	4 MB, the maximum	$895
520-1	1 MB, non-expandable	$169
520-2	socketed, no RAM	$ 79
1040A	1 bank sockets, no RAM	$110
1040B	fully socketed, no RAM	$149
1040C*	2.5 MB + 1 bank sockets	$495
1040D*	4 MB, the maximum	$845
1040K	kit, equivalent of 1040A	$ 68
Clock	option on memory board	$ 30
Clock	stand alone, 520 or 1040	$ 38

All clock upgrades come with software.

Please note that all populated boards - identified with a * after the order number - carry chip adjustments to cover fluctuating DRAM prices. An off 5-25-88 these adjustments are $18 for 1 Mb chips. 16 chips are needed per bank. Pls. phone for current adjustments!

tech-specialities Co.

1022 Hodgkins, Houston, TX 77032
(713) 590-3738, 590-2068

Distributors for
Australia:Tech-Soft, 460 Stirling Hwy, Ste 37 W. Austr. 6011, Tel.: (09) 385-1765
E.-Canada:Computer Country, 148 Waterloo St., Stratford, Ont. N5A 4B4, Tel.: (519) 273-1011
Germany:Ingenieurb. Dipl.-Ing. M. Krompasky Schillerring 19, 7751 Grosswallstadt, (06022) 24405

Atari 520ST, 1040ST fm, 1040STand MEGA are registered trademarks of Atari Corp.

CIRCLE #104 ON READER SERVICE CARD.

Computer Garden

Wilkes-Barre & Scranton's Favorite Computer Dealer

Panasonic Printers

$165

Model 1080i-II, 10" wide dot matrix. Tractor, 144 characters per second. Elite, pica and compressed. Great letter quality & graphics. Epson, Centronics compatible. Fits any computer. 2-year warranty!

Model 1091i-II. As above except 90% faster. $189

Nashua Floppies

Quantity	30	100	200
5.25"SSDD	$19	$55	$107
5.25"DSDD	$22	$58	$113
5.25"DSHD	$39	$109	$215
3.5"SSDD	$39	$109	$215
3.5"DSDD	$46	$139	$275
3.5"DSHD	$144	$439	$869

High Quality.Lifetime warranty.Boxed. Rated best in Byte magazine-9/84

Avatex Modems

1200e: $79 2400: $179

Hayes-compatible. Elegant. 1040ST's: $Call

1040ST prices include testing.

Video Digitizers:

Color Computereyes ST $179
B&W Computereyes XL/XE $59

Scanners:

IMG Scan ST (B&W) $79

ST Accessories:

Monitor Master $44.99
Mouse Master $34.99

Star Printers

NX-1000: $169

NX-1000 Rainbow: $229

Antic

Spectrum 512 $55.99
Cyberstudio $69.99

Intersect

Interlink ST $24.99

Michtron

GFA Basic Interpreter $38.99
GFA Basic Compiler $38.99
GFA Basic Companion $32.99
GFA BASIC Book $24.99

Soft Logik

Publishing Partner $64.99
Publishing Partner Pro. $119
Font Disk #1 $19.99
Font Disk #2 $19.99

Dr T's

The Copyist $Call
KCS (Sequencer) $Call
MIDI Recording Studio $25.99

Timeworks

Wordwriter ST $48.99
Datamanager ST $48.99
Swiftcalc ST $48.99
Partner ST $32.99
Publish ST $Call

Toll-free order line:

1-800-456-5689

For information call 1-717-823-4025
3% charge for VISA-MC-AMEX. Shipping charges extra.
Computer Garden, 106 W. Carey St., Plains PA 18705

CIRCLE #105 ON READER SERVICE CARD.

ALICE

The Personal Pascal

An integrated programming environment with 700 HELP screens, an editor that makes errors impossible, and the best GEM interface anywhere. **Only $79.95.**

"An excellent value." – Antic

"It is about as painless a method of learning Pascal as can be devised short of hypnosis. It works!" – Computer Shopper

"The product is all anyone could ask for. I would recommend this product to anyone who is considering learning PASCAL . . . or anyone who wishes to prototype small applications which deal closely with GEM." – ST Informer

Orders: 1-800-265-2782

Looking Glass Software

124 King St. N. Waterloo, ON. N2J 2X8 519/884-7473

CIRCLE #108 ON READER SERVICE CARD.

The Absen Revolution

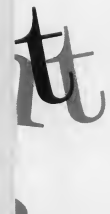
by Mike Donahue

Even the most noble of policies is subject to reality.

He must be a miracle worker. Nearly four years ago, Jack Tramiel, the founder of Commodore, purchased Atari, Warner Communications' "problem child," and borrowed from Warner the $240 million needed to do it. Then in a flurry of activity he emancipated the bloated Atari from all its excesses and extravagances, fired thousands of employees worldwide, consolidated the operations of the 40-building Silicon Valley headquarters into a single two-story building and concentrated all manufacturing in one Taiwan facility. He trimmed Atari to a mere skeleton of its former self, enabling it to weather even the most destructive of financial storms.

But after giving Atari a new lease on life, Tramiel had a rude awakening. The marketplace was vastly different from what he was used to at Commodore. The impact of low price in the low-end computer marketplace was waning, a fact made clear by the very poor sales of the new $89 Atari 65XE home computer. Outside of a price war, Tramiel is like a fish out of water. The name of the game had become marketing superiority, a game which he deplores.

He looks disdainfully upon companies like IBM and Apple that succeed



more through image and advertising than through the merits of their products. "You can't compute with image; you have to compute with a computer," he is fond of saying.

But even the most noble of policies is subject to reality, and Tramiel's bona-fide methods are becoming increasingly less effective as consumers respond more and more to demagogic advertising. Few Atari enthusiasts would deny that in terms of price and technological superiority, the Atari ST is the best personal computer on the market today. For a computer to offer the same technology as the Apple Macintosh at less than half the price is astounding. But to include in the same price color, standard input/output ports, a built-in hard-disk port, and higher resolution graphics, is revolutionary. So, where's the revolution?

The answer lies in the fact that never before has marketing had the profound importance that it has in this decade. In today's marketplace, the proliferation of cutthroat competitors has made it most difficult to sell a superior product unless it is backed by an armada of the most clever and calculated marketing strategies around. With this fact emerging, it can be said that the need to design a clearly superior product stands in the shadow of the need to develop preeminent marketing strategies. Some firms demonstrate such marketing prowess that they can even make a best seller from a product of inferior technology. (Right, IBM?)

Yet, Atari is laboring under the misapprehension that if it simply provides the technologically best product in its class it will impose it on the marketplace. This antiquated "better mousetrap" philosophy no longer holds true in the contemporary American marketplace. If Atari honestly believes the ST is a better product than the Macintosh, it must find it puzzling that the Mac outsells the ST many times over. Apple peddles a cool 60,000 Macintoshes per month. The Atari St's success has been hampered not by technological inferiority, but lackluster marketing.

What's in a name?

At the time of Jack Tramiel's purchase of Atari, many argued that having discarded so much of the company's technological resources, he had essentially paid $240 million for intangible items: the Atari name and symbol. While this is nothing new to the history of business, which is rife with events in which million-dollar transactions involved merely a name or logo (Victor Kiam paid $25 million for little more than the Remington name), this instance stands out. To many, Atari is still thought of as some corporate giant that has enormous strength backing each product. "Atari" is a relatively old name that is in the heads of millions of people all over the world and has possibly more recognition than "Compaq" or "Tandy."

Because Tramiel realized that a start-up computer company would have little chance succeeding in the ferociously competitive marketplace without name recognition, Atari was of prime value to him. But although name recognition is an assessable commodity, Atari's may be the wrong kind for the purposes of Tramiel and sons. The perception as a game company is the single biggest problem facing the company, eclipsing its numerous accomplishments.

The Atari name stands as a roadblock to the lucrative business market. And name problems can't even be eluded in the area where Atari products are the most well received: the music industry. For as the May '87 issue of *ST-Log* revealed, many professional musicians cover up the nameplates with gaffer's tape before bringing their Atari computers to the studio.

Ironically, it was reported in *Infoworld* that Atari was once attempting to garner a defense contract for its upcoming 32-bit computer. Absurd thoughts come to mind when trying to imagine what the outcome might have been if they had succeeded. A startling *Washington Post* headline may have read "U.S. Defense Uses Atari Computer!" And Soviet leader Mikhail Gorbachev might be quoted as saying, "I sincerely hope they aren't using them in their missile silos."

Even in the education market, Atari's name may act as a hindrance. One can imagine that if Atari computers began cropping up in schools, angry PTA members might protest, "I was appalled to learn that our children are using Ataris in the classroom!"

Setting aside the voluminous blunders, one has to admit that the performance of Warner's Atari, Inc. in creating an image of the No. 1 video-game company was irreproachable. Little did Jack Tramiel know that Atari was an inextricable part of an unsavory package deal: With ownership of the company came the onus of undoing what Warner's Atari did so well during the video-game craze. The effects of a successful marketing campaign can last indefinitely and thus, the marketing successes of Atari, Inc. haunt the new Atari Corp. today as it emphasizes computer products. Perhaps many a potential Atari computer sale fizzled when the salesperson's sermon about the superior features was drowned out by the "Have you played Atari today?" jingle playing incessantly in the prospective customer's head.

Indeed, when the company first changed hands, its newly recruited executives expressed a strong desire to alter Atari's image. Sigmund Hartmann, newly assigned head of software, acknowledged in a *Compute!* interview that the old Atari sought to be recognized mainly as a video-game company. "We definitely want to change that," he said. The new emphasis was to be recognized as a computer firm. But its reasoning seems somewhat paradoxical when the company laments its video-game image while it continues to sell video games.

Atari's argument is that it seeks to

The Absent Revolution

capitalize on the fad's reappearance and then break camp when it withers. However, the expense of milking this "cash cow" today is the furtherance of irreparable damage to Atari's image, and the limiting of the company's future as demand continues to circumvent low-end computer markets in favor of vertical markets.

Retired U.S. Navy Rear Admiral Grace Hopper, known as computing's "first lady," once illustrated the power of a name quite well. In an *Information Week* magazine interview, Hopper said, when discussing today's computer power in relation to the past, "We never should have called them microcomputers. Because . . . you can't make anybody believe that they're more powerful than the mainframes were a couple of years ago." As someone who understands the importance of names, Hopper might agree that Atari's is a hindrance even though the company sells computers far more powerful than the school-bus-sized Mark I mainframe she once used in the Navy.

Atari has excellent products. Though not a panacea for all its perplexities, a new image could help Atari gain entrance to many important markets.

All or nothing

An accepted truism among modern marketers is that it is nearly impossible to unseat a champion from the No. 1 position. (Pepsi or Burger King will attest to this.) Given this, an historic event must have just occurred because a No. 1 slot was recently usurped by Nintendo, now the leader in videogames. Atari has become No. 2, followed by the No. 3 Sega. So, it would appear Atari went only halfway in its obstinate pursuit of the resurrected fad, making itself susceptible to a Nintendo coup. Those Japanese who Tramiel is so deathly afraid of outwitted him with technological innovation. (Who would have thought a video-game machine could be equipped with a moving robot?)

It would be in Atari's interest to have a binary position on the video-game market: in or out, on or off, all or nothing. If the video-game arena is the company's chosen domain, it should try vigorously to reclaim its leadership position there. But if the company waits to occupy computer territory, the "nothing" approach would have a far healthier effect on the Atari name. The company has instead opted for the "have cake and eat it too" approach, expecting to dispel its video-game image in the midst of selling video games. The only feasible way for Atari to capitalize on the boom without fueling the video-game connotations is to sell the machines through a subsidiary with a different brand name. If Atari were to select and vigorously promote an imaginative new name, it could pursue the video-game market with unbridled voracity while sheltering its name from further injury.

It would seem like a waste of the Atari name to start completely from scratch in this market with a name void of such recognition. Yet who but the most diehard arcadegoer had heard of Nintendo or Sega? In spite of this vexing problem, Nintendo managed to become the leader in the video-game market, and Sega was able to establish a stronghold.

The equity in the Atari name is obsolete if the market doesn't exist tomorrow. As the video-game fad makes its small, unspectacular comeback, there is no guarantee that it won't deflate again, possibly to extinction. Perhaps the present state of this ephemeral fad can be likened to Halley's comet's second pass through the solar system before it blasts into oblivion.

Home doldrums

The cancer that killed the video-game spread to the home computer as well. Just years ago, Timex was able to peddle oceans of its $100 Timex Sinclair, a desktop calculator-size computer with a 2K memory, rudimentary black-and-white graphics, a flat membrane keyboard, and almost no software. Somewhat indicative of the vitality of the present market, the $89 Atari 65XE computer with a 64K memory, 256 color graphics, full-stroke keyboard, and 2,585 pieces of software is on the verge of being discontinued due to disappointing sales.

Apparently, the home computer, like the video games, was exemplary of computer technology which, by being used for amusement, was turned into a fad item. After curiosity was appeased, the home computer lost its purpose. A computer is a serious tool, just like a hammer. A hammer isn't bought for amusement; it has a purpose, a function. But alas, it's almost as if the home computer bellows, "Give me a

purpose or give me death!"

A *Chain Store Age* trade magazine survey revealed that 76.7% of shoppers did not know what they would do with a home computer if they bought one. Sure, a home computer can balance a checkbook, but automating such a simple task as this for novelty's sake can only overcomplicate and slow the process. A home computer can't make toast, can't clean the rug, can't give you a really close shave, and therefore, a home can survive without one. Maybe all those millions of home computers were purchased solely for game-playing purposes, making them slightly more esteemed video games susceptible to the fad's expiration.

Today, the breadth of home-computer buyer types has sharply narrowed, leaving but a rare species called the hobbyist. Still, Atari maintains that the home computer market is alive and kicking. The company gauges the market's viability by the excitement displayed at Atari enthusiast fairs. By the conventions of Atari 's logic, one would measure America's current interest in Star Trek by the level of enthusiasm at a Star Trek convention.

Perhaps sadder than a company's failure to create new opportunities for itself is its failure to exploit opportunities which already exist.

Word of life

It appears that the only segments of the home market that have truly survived are word processing and education. Even the much-touted "home appliance control" market hasn't inspired much more than a yawn. But the existence of the low-end word-processing market has prompted the creation of the $649 Magnavox Videowriter and similar specialized word-processing machines. Having done their marketing-research homework, the makers of these machines have discovered the demand for a computer dedicated solely to word processing.

While these machines seem like wastes of money to the computer-wise, Their beauty lies in their simplicity: They only do word processing and appeal to the nontechnical consumer who is confused and intimidated by the diversity of functions of a home computer.

With the proper arrangement of peripherals, including a 1027 letter quality printer, the Atari 130XE computer could easily be sold as a word-processing machine. Just recently, Atari found it necessary to sell another computer under a different guise: the new $150 XE video game which is technically identical to Atari's $89 65XE computer. Similar ingenuity can be applied to seize a sizable portion of the low-end word-processing market.

At the realistic price point of $600 for an Atari computer-cum-word processor, Atari could immediately undercut the competition—something Jack Tramiel is notorious for. Moreover, with a 1027 letter-quality printer Atari's word processor would have an obvious advantage over the others that, with the exception of expensive models from Vidco and Brother, have dot-matrix printers.

One Atari reseller has already spotted the new trend: in magazine ads, JS&A (heralding the "Products That Think" slogan) portrays the Atari 130XE as a space-age typewriter that will forever change the face of typing. Apparently, "home computer" just doesn't cut it among consumers any longer. At least they know what they'll use a space-age typewriter for.

A lion's share of the education market—60 %—belongs to Apple. Following Apple is Tandy, with impressive sales of its IBM-compatible Tandy 1000 to educational institutions. And IBM is making another go at this market—the first being with its failed PCjr, a disappointment by IBM-standards—with a somewhat impressive networking PS/2 model 25. Atari would be forced up against the ropes if it tried to claim a share of this market through conventional means.

But innovation transcends even the biggest of marketing dilemmas, allowing the smallest of companies to join the higher ranks without the necessary marketing muscle. Innovation—not superior marketing—allowed Nintendo to become the No. 1 video-game maker. Offering schools something innovative, something that can't be found on competitors' machines, may be Atari's only ticket to schools.

Nintendo *warfare*

To expect Atari to whip up a technical innovation overnight is a bit much. But such expectations are unnecessary, as Atari has such a breakthrough already in its possession. For well over two years it has had a working CD-ROM (Compact-Disk ROM) drive and the necessary system software under wraps. Although many large firms are privy to the technology, the CD-ROM has yet to be unleashed in droves. It is

probably not a consumer item—at least not at its current price—but in schools and other institutions, it could revolutionize research methodology due to its 550-megabyte storage capacity. Elementary schoolchildren could more adequately tap the endless resources of an encyclopedia. The optical drive could even escort Atari into the public library.

But while schools and institutions could pay for the CD-ROM painlessly, Jack Tramiel refuses to disseminate this technology until it meets his price destination. How considerate. But while Tramiel engages Atari in a waiting game, Apple or IBM is free to introduce the CD-ROM, thereby quickly establishing either as the first and foremost source of this technology.

A company is only recognized as a pioneer or innovator if it is first to bring the innovative product to market. And although a unique relationship between Activenture and Atari wrought the first CD-ROM solution, Apple may very well swipe the credit by simply introducing the CD-ROM first. Worse, if Apple does so, it will cement the monopoly it already has on the education market, and its position will become virtually tamper-proof. For Atari, being first with a CD-ROM drive would enable it to leapfrog the murderous competition. In short, the CD-ROM is the key to permeating the barriers that have prevented Atari from having a stake in the education market.

Today, the breadth of home-computer buyer types has sharply narrowed, leaving but a rare species called the hobbyist.

Beautiful music

A brilliant aspect of the Atari ST's design was the inclusion of MIDI (Musical Instrument Digital Interface) ports as standard equipment. Virtually every synthesizer on the market has MIDI ports. And until recently, while countless MIDI synthesizers existed, the missing link—the computer to conduct the peripherals—had yet to appear. Now, with the exception of the obscure Yamaha CX5M, the Atari ST is the only MIDI computer on the market.

Interestingly, an Amiga, IBM PC, and even a Macintosh can be configured with MIDI ports. Apparently a computer with a feature that exists as standard equipment has the advantage of appearing somehow more suited to the appropriate task. So, by virtue of being MIDI equipped, the Atari ST has the immediate upper hand in the music market. The ST has, in fact, been greeted with excitement by musicians.

Yet the computer was discovered not through advertising, but through the desperate quest by musicians for a MIDI computer. All ST advertising that targets musicians has been done in software ads by Hybrid Arts, a MIDI software firm. Atari has yet to advertise autonomously. While it may seem convenient and economical to be able to virtually ignore a market and still enjoy success there, the full capacity for success can't be realized without active pursuit. And if this market's potential were to attract any noteworthy competition, Atari might not be able to hold its ground after failing to minimize danger by establishing an identity as the foremost MIDI computer supplier.

In the new age of marketing, called the "positioning era" by Trout & Ries, Inc., a New York advertising firm, the most secure place for a company is the No. 1 position in a given market—even a niche. Yet Atari is adamantly opposed to pursuing niche markets on the belief that a company becomes trapped in such markets and loses its ability to compete effectively in broader ones. Atari's "claustrophobia" is unwarranted—even IBM captures niche markets.

An uncontested niche poses an opportunity to claim a very large piece of a small pie. But in the case of the MIDI market, the pie might not be all that small. According to the *New York Times*, sales of computers and software for MIDI uses recently topped $500 million. So the MIDI niche may be booming bigger than the video-game market. Perhaps sadder than a company's failure to create new opportunities for itself is its failure to exploit opportunities which already exist. MIDI is one such opportunity.

Apple too

It's comforting to note that Atari doesn't have a corner on the market for image problems. Tandy is currently hog-tied by a home computer/Radio Shack image as it tries—so far unsuccessfully—to crack the business market. And Apple, of all companies, has found gaining space on the corporate desktop to be a Herculean task. It seems that Apple computers were recognized for their place in homes and schools, not businesses. An attempt to make the Appfe name denote "business" in addition to "home" would be futile. The public will remember Apple's original identity for a long time to come.

Before marketing whiz John Scul-

ley's arrival, Apple had been hammering away, trying to penetrate the business market with the Macintosh computer. Sculley then managed to help the Macintosh overcome the ill effects of Apple's unbecoming image. The word "Apple" is noticeably absent from current Macintosh commercials. The concept being sold in these ads is that of "Macintosh," a word without preconceptions.

Which does a Crest commercial sell: the name Crest or its maker's name, Proctor & Gamble? Is a Sanka commercial selling the name Sanka or General Foods Corp. the name of its maker? The same technique is being employed to disassociate Apple from Macintosh.

Meanwhile, Apple's image is profiting from the Macintosh (an interesting contrast to the classic situation). The results have been phenomenal: Apple is now the No. 2 personal computer maker after IBM (with help from desktop publishing). Even the IBM clone makers follow Apple, which became No. 2 two without any help from IBM compatibility.

While it has not yet attained notoriety as a computer company, Atari is making overtures to corporations. Software president Sigmund Hartmann has been assigned the laborious task of negotiating sales of Atari STs to Fortune 1000 companies. Perhaps to Mr. Hartmann's chagrin, many a nearly consummated sale has dissolved when an executive with buying power caught a glimpse of an Atari video game in a Zayre flyer (to say nothing of a Toys 'R' Us flyer).

But alas, Jack Tramiel isn't truly comfortable selling to the business sector. He is most at home, at home. In his forward to an ST Logo tutorial, he intimated his devotion to making the highest possible technology available to the average home by means of vigorous cost reduction. There's almost an air of benevolence about Mr. Tramiel as he pursues this noble obsession.

While it's not fun to critique someone's *raison d'etre*, it's hard not to wonder about the future. If Tramiel continues on his present course, will he eventually bring the price of a Cray supercomputer to within the family budget? A tantalizing, albeit sarcastic thought to ponder until better reasoning provokes a pertinent query: How many homes are in demand of the power of a Cray supercomputer?

No matter how good a bargain is, families won't buy something they can't determine a use for. Businesses, on the other hand, will always crave the most advanced computer technology and can use it constructively. Therefore, unlike the home market, the future existence of the business market is a certainty. With the home market's loss of viability, Atari's acceptance into the office is not a mere ideal, but a necessity for survival. Atari's future could hinge on whether it can establish a reputation as a computer maker today.

The business-computer market is driven more by need than by want. Unlike consumers, businesspeople can ill afford the luxury of being fickle. Ask a typical businessman to imagine a day in which all the office's personal computers were broken down. He will, undoubtedly, paint a grim picture that makes their necessity more apparent. Businesses depend on computers. The need for business PCs is as lasting as the need for automobiles.

Full service

Naturally, customer satisfaction is pivotal to success. And indeed, Jack Tramiel believes that buyer satisfaction today ensures repeat customers in the future. In a 1985 interview with *ANALOG Computing*, he said, "Intelligent companies are going after the long term—not to cheat the customer, not to be greedy."

But here surfaces a pertinent issue. Atari has often been attacked for its poor service and repair provisions. Customer satisfaction certainly is affected by the facility of getting the computer fixed when necessary. This is especially true in a business setting, where productivity grinds to a halt when computers break down. Under Jack Tramiel, Commodore was notorious for its watery commitment to servicing its computers. Apparently, when Tramiel said, "Atari will become no different from the old Commodore," he intended to be taken literally.

Long-term strategy is vitally important to future growth. But a dangerous assumption can be made when a trend is misread. In a recent *Infoworld* article, Tandy's Graham Beachum, then president of Computer Merchandising, outlined Tandy's strategy for penetrating the business market. Beachum, who recently defected to Dell Computer, claimed that a buyer of a Tandy home computer will, without doubt, return to Tandy if the need for an office

Though not a panacea for all its perplexities, a new image could help Atari gain entrance to many important markets.

computer arises.

His logic is invalidated by consumer behavior. The move from a home computer to a business computer is not seamless, as the consumer mind seems to draw a clear distinction between the two types. When a home computer owner decides to "get serious" by investing in a business computer, he'll turn to a different company, one that is indigenous to the business milieu.

IBM made the opposite, albeit equally fatal assumption with its PCjr home computer. While it accounted for only 1% of total revenues, the PCjr might sell a few mainframe computers a decade or so down the road, thought IBM. But the hypothesis proved untestable as poor junior never really made it into the home. Why? Because, as acknowledged by Al Ries and Jack Trout in their book *Marketing Warfare* (McGraw-Hill, 1986) IBM is renowned not for its place in the home, but in the office. Apple and Commodore laid claim to the home.

Atari, under a similar illusion, believes that all the youngsters who grew up on its video games are now mature enough to want computers and will stick with Atari. Once again, the fly in the ointment is Atari Warner's splendid job of establishing Atari as the first name in videogames.

"Atari" actually became synonymous with "videogame." This potentially desirable effect occurs when a product assumes its closely associated brand name: "Xerox" became a surrogate for "photocopy"; "Jell-O" substituted for "gelatin"; "Band-Aid" for "plastic bandage"; "Kleenex" for "tissues"; "Vaseline" for "petroleum jelly." Here the brand name becomes generic.

During the video-game craze, kids began to refer to video-game machines in general as "Ataris." A distinction wasn't even made in reference to video-game units of other manufacturers! While this generic brand name may have been Atari's richest asset during the heyday of the video-game craze, it haunts the company today as it strives to sell personal computers.

The logic that says that kids will graduate from Atari video games to Atari computers is flawed. They will graduate from "Ataris" to computers. They'll turn to a totally different company to provide a computer than they turned to for video games. Again, until the Atari's name denotes computers, the company will continue to have little impact on the computer industry.

An equally feeble belief Atari clings tenaciously to is that a new generation of kids is emerging that hasn't yet been exposed to video games, and will foster a new craze reminiscent of 1982. But, unfortunately, this false hope has about as much validity as one that insists that today's Americans will soon be listening to Rossini because they are of a new generation that hasn't yet experienced classical music. Times change.

While it is fun to critique someone's raison d'etre, it's hard not to wonder about the future.

Foot in the door

Given that the demand for business computers shall exist as long as businesses exist, Atari does have reason to set its sights on the office. To get there will take exceptional products, exceptional marketing strategies, and most importantly, a reputation as a computer maker. Atari has one essential ingredient: exceptional products.

The new IBM compatible Atari PC, if properly marketed, might help pry open the corporate door. Hopefully, Atari views this product as a part of a strategy, not just another way to make a fast buck. If IBM compatibility gets Atari on the corporate purchase order, it may assist in the ushering in of a new identity for the company.

But Atari stumbled when it decided the Atari PC would be sold through mass-market channels. The company should follow the paradigm of firms like Apple, IBM, Zenith, Compaq, and others who know that computers simply don't belong in the mass market where they are cheapened by the environment and are considered inferior to computers sold in specialty stores.

There exists the unfounded belief that the $680 Hyundai mass-market IBM PC compatible is selling like insect repellent in a nudist colony. "Like air conditioners in Alaska" is a more befitting analogy because as a recent *Business Week* article, "Hyundai Computers Are Stuck in the Slow Lane," reveals, the case is very different. While a slew of them made their way through the distribution channels to retailers, few made it off the shelves. The article refers expressly to mass marketing as the malefactor in the Hyundai PC's failure and notes that "Customers found computers stacked alongside microwave ovens with no trained sales help." An *Advertising Age* article described the situation similarly, noting that one large chain has al-

ready dropped the Blue Chip due to slow sales.

The specialty store is the personal computer's most suitable habitat. There, personal selling can occur, and if an advertisement hasn't enlightened or persuaded a consumer enough, a knowledgeable salesperson can.

XF4Q1T19?

Is the Atari ST destined to have only a cult following, or is it merely a sleeping beauty that has yet to awaken to the world? Many attribute the computer's relative anonymity to Atari's failure to get the word out through heavy promotion. But the unprecedented competitive climate of today's American marketplace makes effective advertising a gargantuan feat, for the barrage of advertised messages comprises an indecipherable noise. If the message isn't potent enough, it only adds to the noise.

So it is no longer enough to simply "go tell it on the mountain." Such conditions amplify the need to choose product names that are highly distinctive. Apple chooses names like Unidisk, Laserwriter, Macintosh, Imagewriter LQ, etc. Atari, on the other hand, selects names like XM301, SF314, SM804, XF551, SM1224, 1040ST, SF354, etc. This cryptic nomenclature makes effective camouflage, but is counterproductive to the cause of fighting product anonymity. It also goes against the grain of the intuitive user interface which was designed to make the computer more appealing to the "non-techie." "Computers for the masses, not the classes," Jack Tramiel always says.

Moreover, people are more inclined to discuss products whose names they can remember. Univac, Cray, Domain, VAX and Macintosh are names that have commanded attention in the computer industry. Even the name Amiga is more widely recognized than 520ST and 1040ST. The Amiga reportedly sells as well as the Atari STs, despite the vast difference in price.

Megamania

Judging by the use of the name Mega for the new Atari computers, the trend may be changing. But while the use of this metric prefix could mark the beginning of a much-welcomed new style, the name is a bit too tacky and toyish sounding for the serious machine it represents. Mega sounds more like a new Hasbro Transformer toy than a $2,000 to $3,000 computer.

It's a shame that buyers often judge books by their covers, but with the plethora of computers on the market, realizing all distinctive features of all models is impossible. It has become far easier to judge a computer product by cursory examination of seemingly trivial aspects. Given this, the Mega ST with its tacky name may have begun life with an immediate handicap.

Interestingly, with the announcement of the Mega series ten months premature of the delivery date, Atari felt the symptoms of Osborne syndrome in the form of a drop in sales. Evidently, Atari didn't take a lesson from Adam Osborne's classic act of folly. When Osborne Computer hastily announced the Osborne 2 portable before the machine was ready, an acute drop in sales of the Osborne 1 sent the company right into the depths of Chapter 11 bankruptcy.

Publicity—the most cost-effective form of advertising—was apparently sought when Atari released pictures of its new Mega machines to the press shortly after the computers were announced. But unfortunately, the Mega was shown displaying a Neochrome picture of a cartoon-like robot, hardly reflecting the awesome power of a four-megabyte computer selling for $2,500.

Apple's photos of its new one-megabyte Macintosh II computer showed the machine displaying a fabulous ray-tracing demo in which the Apple logo was reflected in several chrome balls. A very similar demo, Shiny Bubbles, existed for the ST computers long before the appearance of the Apple demo. In fact, the two are so similar that one would swear that the Apple programmer had modeled the Mac II demo after Atari's. Now, due to the Apple demo's wide exposure, the opposite might be assumed.

Apollo also uses ray-tracing screens in ads to show off the power of its CAD workstations. Atari, on the other hand, rejected a ray-tracing demo of enormous potential in favor of a cartoon robot. In doing so, the company procured but one thing: very poor publicity. To appreciate the weight first impressions carry is to realize the detriment that may have been induced by Atari's photos.

Still, the Mega is a refined machine with many welcome additions, including a detached keyboard. It was IBM that trained the consumer to believe that only a computer with a detached keyboard could bask in the esteem of a true business machine. The two-piece computer was thought superior to the one-piece computer, notwithstanding all technical differences. The one-piece configuration had forever become the mark of a home computer.

Another factor contributing to computer product's image is visual impact. Ideally, a high-capacity hard-disk's visual appearance should cry out, "Power! Capacity!" But due to the colorful, tacky lettering, the Atari SH204 20-megabyte hard disk's visual appearance screams "Fisher-Price!" Fortunately, the new companion hard-disk to the Mega computer exhibits a marked improvement in visual impact.

The evolving quill

Sadly, Atari may never have such a unique opportunity as the one Apple had to unlock the door to the business world. Desktop publishing, which Apple merely stumbled upon, was the vehicle for getting a "foot in the door" of the corporate office.

At first, despite the Macintosh's dazzle, managers simply weren't given enough cause to switch from IBM. The IBM PC and compatibles could handle all the applications that the Mac could handle, and there was no reason to change, especially given that MS-DOS was the looming standard. Enter desktop publishing. Finally, there existed a function uncommon to the IBM PC and all other MS-DOS machines.

The Mac had become the only route to this new application. As a result, the corporate back door was pried ajar, and the Macs began scampering in.

Today, desktop publishing can be perfomed quite easily on an IBM PC or clone, but the Macintosh remains the leader in desktop publishing, having established itself there first.

A large portion of Apple's $2.4 billion in sales in fiscal 1987 came from the desktop publishing market, often discounted as a mere niche. Dataquest, a market research firm, believes that by 1990 desktop publishing will be a $4.9 billion industry.

Atari, though a little late, is vying for a piece of the action by introducing its own desktop publishing system. *With* pricing being by far the most prevalent form of competition, Atari will sell its two-megabyte Mega ST computer and

laser printer package for a competitive $2999—about $2000 less than Apple gets for its laser printer alone. Because of Atari, the desktop publishing phenomenon may be felt more widely as it reaches those with tighter purse strings.

But consumers mustn't be given the impression that Atari's desktop publishing system is simply as good as Apple's for less money. Greater incentive is required to lure buyers away from the leader. Advertising for Atari's publishing system must be focused on features that are unique amidst other systems' features.

Better, weaknesses in the Macintosh should be exploited petulantly in Atari's advertising. Suitable ads should magnify the deficiencies of the Mac system that are corrected by the Atari desktop publishing system. (The ads could also employ a sly, but effective tactic by referring to the Macintosh as the Apple Macintosh, thus reopening the wounds Apple is healing disassociating Apple and Macintosh.) The most obvious and complained-about flaw of the Macintosh is its tiny screen. No trip back to the Atari drawing board is necessary because all ST models have screens that improve on the size of the Mac's.

Ironically, the Macintosh's strongest feature—graphics—becomes a weakness when compared to the Mega ST. The screen resolution of the Macintosh is 512 x 342 pixels. When compared to the ST's monochrome resolution of 640 x 400, the Mac's is shy of 128 x 58 pixels. Voila!—another Macintosh vulnerability is born. While Apple's ads tout the Mac's WYSIWYG (What You See Is What You Get, or the ability of the computer display to approximate the final printed document) ability, Atari's ads can justifiably claim that "the Mega computer is 'WYSIWYGier' than the Apple Macintosh."

As color separation becomes of more concern to desktop publishers (and perhaps becomes amended to the definition of WYSIWYG), the Mega ST will be ready with its built-in color capability and optional color monitor. Still another weakness is unearthed because the Macintosh was not designed to handle color.

In Macintosh weakness lies Mega strength. But the factors that constitute Macintosh vulnerability won't be in place very much longer. Ergo, opportunity knocks.

The difference is graphics

As noted earlier, contemporary marketing says that changing the consumer's initial view of a company is nearly impossible. So direct attempts to make Atari mean "business computers" overnight would be fruitless. Is this predicament hopeless?

Not entirely. What the consumer mind will acquiesce is something already known: Atari means "computer graphics." This broader notion is fully compatible with the public's view of Atari as a video-game company, yet it establishes that Atari makes computers. If Atari's advertising were to repeat this accepted message over and over, graphics may eventually become the primary connotation of Atari's name. And while earning a reputation as a computer firm, Atari would also be differentiating itself from its competitors.

Additionally, as computer applications become increasingly focused on graphics—as are computer-aided design and desktop publishing—Atari's name may actually be an asset. For the first time, Jack Tramiel's $240-million investment in the Atari name and symbol will have paid off nicely. "Atari: the difference is graphics."

The prophet speaketh

What does the future have in store for Atari, the 15-year-old enterprise that once employed Steven Jobs and Stephan Wozniak before the two founded Apple Computers? Newly elected senator Timothy Wirth of Colorado is labeled a typical "Atari Democrat," a person, *Time* magazine reports, "who urges growth and investment in high-technology industries."

Is the name used entirely with sarcasm here or is this a telling clue that Atari may be an icon for high technology? Does a peephole to Atari's future lie in the British post-punk band Sigue Sigue Sputnik's song "Atari Baby," about life in a future high-tech world? Or will time realize the prophecy of the movie *Blade Runner*? In this film set deep in the future, colossal city billboards everywhere displayed giant Atari ads as if the company had erupted into a monstrous, domineering corporate empire. Could this scenario be an emblem of Atari's future? No. At least not unless Atari ceases to mean video game.

More troubling still, back here in the days of Iran antics and Oliver North, Atari struggles against a deluge of opposing factors: A home-computer hysteria has abated following the video-game exodus; a business market is an intense battle of heavyweights like IBM, Apple and Compaq; a lucrative education market is dominated by the Apple-Tandy faction and may soon play host to IBM with its mighty resources, advertising capital and networking PS/2 model 25; a burgeoning desktop publishing market claimed largely by Apple is awash with IBM clone systems and will be further drained of its resources by IBM's upcoming PS/2 desktop publishing system.

So where does this leave Atari? Fortunately, the prospects are not entirely bleak for the company. Atari can be No. 1 in the MIDI market which it has entirely to itself. Many believe that today's computer game is won through software. With powerful, established software products like *SBT Database Accounting Library*, *WordPerfect*, *dBMAN*, *Drafix CAD*, *DAC Easy Accounting*, *Microsoft Write* and *Ready, Set, Go!*, Atari has prolific fodder. This software must be exploited in Atari's ads, or its potential for selling STs will be wasted.

Yet tomorrow's computer game may be won differently. It may be won in networking, multitasking, internal expandability, connectivity, or brand-name partiality. And as the current competitive trend progresses, success will grow even more dependent upon marketing. If Atari is to outrun adversity and reach its outspoken goal of becoming a $1 billion operation by 1990 it better get busy, for it has a tremendous amount of work to do. The loss of Atari as a contender in the computer market would be tragic. Atari is the computer manufacturer with the most equitable cause of all: to give the buyer, whoever he may be, the best value for his money.

Mike Donahue, *a 20-year-old student living in Vermont, considers himself a "Vintage Atarian," having been obsessed with Atari for nearly six years. Currently, he uses a monochrome 1040ST and a Hewlett-Packard LaserJet Series II to run a small typesetter and graphic design firm. He welcomes reactions to the article and can be contacted through* ***ST-Log*** *Magazine.*

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $105

SAVE TIME AND MONEY SUBSCRIBE TO ANALOG

SAVE $14 OFF THE COVER PRICE WITH THE CONVENIENCE OF HAVING ANALOG DELIVERED DIRECTLY TO YOUR DOOR *BEFORE* IT EVEN HITS THE NEWSSTANDS. GET THE MOST OUT OF YOUR COMPUTER.

SUBSCRIBE TO ANALOG TODAY

☐ 1 YEAR @ $28 — SAVE $14! MCFYY
FOREIGN — ADD $7 PER YEAR
☐ 1 YEAR WITH DISK @ $105 DCFYY
FOREIGN — ADD $15 PER YEAR

☐ PAYMENT ENCLOSED ☐ BILL ME
CHARGE MY: ☐ VISA ☐ MC #_____

EXPIRATION DATE SIGNATURE

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

NAME

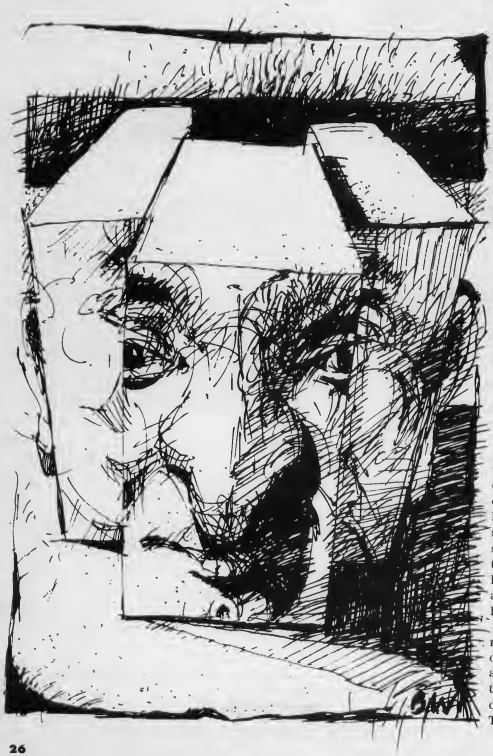
ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16927, N. Hollywood, CA 91615. Offer expires August 31, 1988. Your first issue will arrive in 6 TO 8 weeks.

WATCH FOR IT!

GRAPHICS



3D or Not 3D

A Look at Graphics in ST Gameware

by Andy Eddy

Computer games have come a long way since the early **Pong** days. This evolution, beginning with coin-operated arcade contests, has brought home computer playware to a level that still garners a major share of products sold. But looking over the period of time—a short lifespan, to say the least—that computer games have been around, it's obvious that people's tastes lean towards impatience.

These days there are a ton of companies creating gameware for personal computers; some have been around for a while, and others are simply spare-time endeavors sparking garage-based development of home-brewed products. To be successful, the bottom line is that

you have to keep moving ahead, bringing about new features to pique the end-users' attentions. Anything short of that will spell disaster to the company's long-term survival.

But it's gotten to a stage where everyone who comes out with an entertainment product appears to have the phrase "3-D graphics" attached to it. We'll see how that may be something of a misstatement, as well as how developments in entertainment software have brought us to the present era.

In the Beginning. . . .

While the layout of the game itself holds a lot of weight in a product's market success, you can boil most of them down into similar categories of structure: shootouts (either space- or earth-bound), driving contests (sometimes mixed with the shooting genre), traditional board games, adventures and sports. Predominately, what sets one piece of software apart from the next is how it's graphically represented. Even these can be broken down into categories, geometrically speaking.

The first games to appear on the scene were what we'll call a two-dimensional playfield with one-dimensional movement; **Pong** and **Space Invaders** are examples of this basic layout. Simple left-right or up-down movement—limited to one axis, if you refer to the screen as an X-Y graph—is all that you can manage.

The next generation brought us games like **Asteroids** and **Missile Command**. These went a step further by allowing two-dimensional movement in a two-dimensional playfield. Your on-screen counterpart isn't limited in its range of movement—both X and Y axes can be traveled freely. In fact, Asteroids takes it a jump beyond this with its "wraparound" universe. In other words, the borders of the screen were no longer like a wall to the player. This succeeded in adding more tension and required stronger peripheral vision from the player.

Lastly, the final leap in programming is what we'll term three-dimensional movement in a three-dimensional playfield. Of course, it must be explained for clarity's sake that this three-dimensional universe—using the X, Y and Z (depth) axes at one—is sensory only. Though we'll mention the *few* deviations to the rule in a moment, the games we're talking about now are "simulated" in their 3-D effect, simply because a standard monitor screen is only two-dimensional. "2-D plus" or "2½-D" are more accurate ways of terming these graphic representations.

There are two different ways of creating the perception of 3-D in a 2-D medium. Looking at a TV program shows the first way: The program was filmed on a 3-D stage, and the viewer relates to the perspective of objects on the screen and and how they interact with each other (i.e., size relationships and whether objects move in front of or behind others).

It must be explained for clarity's sake that this three-dimensional universe is sensory only.

There are many good examples of the success of this method, but perhaps the easiest way to tie this in with computer gaming is *Player/Missile Graphics*. Through this, you can set players, missiles and playfields, as well as program how they interact with each other. In essence, this creates a situation similar to the TV show example above, giving the "illusion" of depth.

Space Quest and others in the Sierra On-Line "3-D Animated Adventure" series demonstrate a similar use of this type of process to create the imagery on-screen. They're called 3-D adventures, but you're perceiving the effect because the onscreen characters can move in front and in back of, jump over and crawl under objects, as well as climb fences and trees, or fall into holes, just like television.

The second method was first displayed in **Battlezone**, that point-of-view (POV) display. You get the feeling that you're actually sitting in the vehicle (you don't, as in other games, see your character onscreen), and all objects appear accurately scaled in size to the distance you are from them. Also, as you move around an object, your perspective changes so you can see a different side of that object.

This brought about a breakthrough of sorts. Previous contests depicted your character on the screen (as well as those characters around you), and you subsequently controlled it in accordance with the threats you saw on the screen. POV games give you a whole different feedback to react to, as you can't see all that's around you unless you manipulate your character into a position that allows that view.

To bring this up to date, games like **Starglider** (Firebird Licensees) and **ST-Wars** (Miles Computing) take the basic line-drawing structure of Battlezone and give the user a strong dose of seemingly realistic motion and action. As an avid player of entertainment software, I know that the best test of a product's 3-D imagery is to note how you react to playing the game; in my case, I find that the docking silos of Starglider have me ducking to avoid hitting my head on top. Sure, I know it's a game, but in the heat of battle I find myself getting into it as if it were real life.

On the Horizon

Given the memory size/capability and processing speed of the ST—to be further enhanced by the blitter when it becomes available—vector graphics can give way to solid construction of on-screen objects, as witnessed by ST-Wars and **Harrier Strike Mission** (also by Miles). This spells the future of computer gaming, as the last few years have brought us great leaps in graphic technology. I'm sure that the next few years will be similar, especially if recent creations dictate the path of progress along those lines.

The strides that computer manufacturers and software developers have taken to give us the latest and greatest

3D or Not 3D

has resulted in some pretty wild concoctions. For realism, old and new 3-D technology has been utilized in various manners to that end.

In the old category, you can look back as far as the '40s and '50s to the introduction of two-color 3-D movies. While they fell out of favor, particularly when an increase in technology in this field brought about the advent of color 3-D movies (using polarized lenses), a resurgence of the two-color effect came about recently as some of these ancient films were shown on TV stations across the country, giving viewers at home what they previously had to travel to theaters to see.

Bringing that up to date, **Wanderer** (Pyramide Software, marketed in the U.S. by Paradox) utilizes this two-color effect to bring about a realistic vector-graphic space shootout with depth. Again, the test of this was when I was demonstrating the game for a few friends. When one of the enemy ships dipped off the bottom of the screen, a couple of us tried to look "into" the monitor as if it were a window, thinking that by moving our perspective, we'd be able to see the object below the field of vision that the screen provided. For those interested in experiencing this effect, there is a downloadable demo version of Wanderer in the "Applications for the ST" section of the Analog SIG on Delphi.

It's quite simple to build this effect. Two distinct images are put on the screen, one in red and the other in blue. Putting on a pair of glasses (with one red lens and one blue lens) gives each eye a different view by virtue of the lenses filtering out the image of their respective color. If the two views are properly shifted and the color matching between the glasses and image is good, the brain assembles the incoming data where it is perceived as a realistic 3-D image. The limitations to this technique are sufficient, though. You're restricted on-screen to using only the two colors that are instrumental in forming the effect, and eyestrain can be fairly heavy with long gaming sessions.

Now Tektronix, a strong, high-end graphics terminal producer, through their LC Technologies division, has brought a new twist to the creation of realistic 3-D or, more aptly described, stereo graphics. The **StereoTek** glasses, as they are called, use high-speed liquid-crystal shutters (LCS) as lenses encased in plastic frames. The benefit to this is that similar to liquid crystal watches and video displays, when they aren't activated, the lenses are basically transparent, which is less annoying to your eyes. When they are under the ST's control with compatible software, the stereo effect can be amazing.

The strides that the computer manufacturers and software developers have taken to give us the latest and greatest has resulted in some wild concoctions.

Of course, this article is dealing with the proliferation of 3-D entertainment software, and the potential for that is great. At the time of this writing (mid-June), Antic already had a StereoTek-compatible version of Wanderer ready for release, as well as a Space Invaders/Galaxians-type of contest called **Shoot-The-Moon** in nearly completed form.

On the third-party software front, Shelbourne Software (makers of **ST Pool**) were displaying a beta version of their upcoming **3-D Breakthru** at the Summer CES (Consumer Electronics Show). Slated for September release, this game is a variation of the original **Breakout** with one exception: Your paddle hits the ball "into" the monitor at a wall of bricks at the end of a hallway. To open the product to as many people as possible, the game will have a function key toggle for switching between normal and stereo play modes. The game is stunning in both modes—especially in stereo—and should be popular, laying the groundwork for Shelbourne's name becoming better known to Atari users. They also stated to me their intentions to support StereoTek users with future releases.

Along with Shelbourne's efforts, Gary Yost, marketing director for The Catalog, informed me that Rainbird Software was almost finished with a conversion of StarGlider to the glasses. Though I haven't yet seen a pre-release of the new version, Yost confirmed that it may lose a portion of its original innovative effects—like the memory-hogging, digitized music at the game's start—to allow the cramming of the stereo routines into a standard ST.

In addition, the ever-present programming wizard, Tom Hudson, was one of the first to have his hands on this product outside the Tektronix staff. Among other things, he plans to convert his popular **Livewire** game (a Tempest-like contest for the 8-bit line originally published in ANALOG's issue 12), to the ST and have it playable with the StereoTek glasses. Other manufacturers are looking into using this innovative product with their future game development, depending on how the glasses sell. If the current sales indicate anything—Yost claims over 500 pairs were sold in the first month they were offered—more developers will be following suit.

This technology takes gameware years into the future, all the while making it easier for programmers to utilize it in their efforts. And with the ST in the forefront of graphics creation, you can bet that we in the Atari community will be among the first to witness any other groundbreaking innovations.

Mouse-Ka-source

Personalize your programs! Turns Mouse-Ka-Mania data files into source code for GFA Basic, C, or assembly program.

by Charles F. Johnson

One way to give your programs a distinctive look is to use a custom-designed mouse cursor for some or all functions. **Mouse-Ka-Source** will take a mouse data file saved with my **Mouse-Ka-Mania** accessory (from *ST-Log 18*) and convert it into source code that can be easily included into a **GFA BASIC**, C, or assembly language program.

To use **Mouse-Ka-Source**, load GFA Basic, type in Listing 1, and save it to disk. Then simply run the program and follow the prompts it gives you. If you don't have GFA Basic, a compiled version of the program that can be run without the GFA interpreter is included on this month's disk edition of *ST-Log*, or on the ST-Log ST SIG on Delphi.

If you do program in GFA Basic, this program is especially easy to use; the file that it creates is a full subroutine that does all the work for you. Just merge it with your program and add the statement **Gosub My_mouse** anytime you wish to change the mouse cursor to your custom shape. The subroutine written by **Mouse-Ka-Source** uses only local variables, so it shouldn't interfere with any part of the program you merge it with.

The C conversion routine writes the mouse data as an integer array with the label **MY_MOUSE** (note the capital letters). There are several ways to change the mouse shape from C or assembly; so I didn't write a full subroutine for these conversions. In C, the AES call to change the mouse might look like:

```
gr_moret = graf_mouse
(255,MY_MOUSE);
```

or, alternatively, the VDI call:

```
vsc_form(handle,MY_MOUSE);
```

When you convert a **Mouse-ka-Mania** data file to assembly source, it is written as a series of declare statements (dc.w). Asterisks are used to indicate comment lines, as per the Devloper's Kit AS68 assembler convention.

I think you'll find that **Mouse-Ka-Source** is very easy to use. GEM alert boxes are used to select all the program options, and the GEM file selector is used to choose a mouse data file. The converted source code file will be written to the same directory the data file is in, and the source code filename will be the same as the data filename, with an extension of ".LST," ".H," or ".S" for Basic, C, or assembly source files.

Mouse data files must have an extension of ".DAT" to show up in **Mouse-Ka-Source**'s default File Selector. If you've saved a data file under another name, you'll have to either type it in on the filename line, or edit the directory line to show files with a different extension.

If the mouse data file isn't exactly 74 bytes long (the size of a **Mouse-Ka-Mania** file) the program will display an alert box telling you, "This is not a Mouse-Ka-Mania data file!" A possible problem arises here, if you've downloaded a data file using Xmodem, and it's been padded out to a multiple of 128 bytes. If you somehow end up with a mouse file that isn't 74 bytes long, just load it into **Mouse-Ka-Mania** and re-save it—the length will now be correct.

Charles F. Johnson *is a professional musician, and now semi-professional computer programmer/reviewer/author. He lives in Los Angeles with his wife Patty and Spike, the world's most intelligent (and gluttonous) cat. Charles is a sysop on the Analog Atari SIG on Delphi; his user name is CFJ.*

```
'
' ==========================
'      Mouse-Ka-Source
'   by Charles F. Johnson
' ==========================
'
Wrong$="This is not a|Mouse-Ka-Mania|data file!"
'
RX=-(Xbios(4)<2)
Deftext 1,5,0,32-19*RX
Text 238+12*RX,32-16*RX,26*RX,"ST"
Deftext 1,5,0,13-7*RX
Text 270,32-16*RX,42,"-Log"
Deftext 1,0,0,6-2*RX
Text 320,32-16*RX,"Presents:"
Deftext 1,1,0,32-19*RX
Text 182+4*RX,80-40*RX,272-16*RX,"Mouse-Ka-Source!"
Deftext 1,0,0,13-7*RX
Text 236-4*RX,100-50*RX,"by Charles F. Johnson"
Deftext 1,0,0,4
Text 177,128-64*RX,"Copyright 1987 Little Green Footballs Software"
Alert 0,"  Written in GFA Basic    | ",1," Go! | Exit ",X
If X=2
  End
Endif
'
Drive=Gemdos(25)+65
Ms_path$=Chr$(Drive)+":\*.DAT"
'
Finished=0
While Not Finished
  Cls
  Deftext 1,0,0,13-7*RX
  Text 176,40-20*RX,"Choose a mouse data file to convert."
  Fileselect Ms_path$,B$,A$
  Cls
  If A$=""
    End
  Endif
  If Right$(A$)="\"
    End
  Endif
  For I%=Len(A$) Downto 1
    If Mid$(A$,I%,1)="\" Then
      Ms_path$=Mid$(A$,1,I%)+"*.DAT"
      I%=1
    Endif
  Next I%
  B$=A$
  While Instr(B$,"\")<>0
    B%=Len(B$)-Instr(B$,"\")
    B$=Right$(B$,B%)
  Wend
  '
  If Not Exist(A$)
    Alert 3,"Filename not found!| ",1," Again | Abort ",X
    If X=2
```

```
      End
    Endif
  Else
    '
    Mm$=A$
    DatposX=Instr(Mm$,".DAT")
    '
    If DatposX=0
      Alert 1,Wrong$,1," Oops! ",X
    Else
      '
      Open "I",#1,A$
      '
      If Lof(#1)<>74
        Alert 1,Wrong$,1," Oops! ",X
        Close #1
      Else
        '
        Alert 2,"Which type of file would|you like to create?| ",0,"Basic|C|Asse
mbly",Choice
        '
        On Choice Gosub Bas_file,C_file,Asm_file
        '
        Alert 2,"Process another mouse?| ",1," Yes | No ",X
        If X=2
          End
        Endif
        '
      Endif
    Endif
  Endif
Wend
'
' ---------------------- The conversion subroutines ------------------------
'
Procedure Bas_file
  Mm$=Left$(Mm$,DatposX-1)+".LST"
  Open "O",#2,Mm$
  Print #2,""
  Print #2,"' Mouse Definition Subroutine"
  Print #2,""
  Print #2,"Procedure My_mouse"
  Print #2,"  Local Ix,AX,Mm$"
  Print #2,"  Restore Mm_data"
  Print #2,"  Mm$="";Chr$(34);Chr$(34)
  Print #2,"  For Ix=1 to 37"
  Print #2,"    Read AX"
  Print #2,"    Mm$=Mm$+MKI$(AX)"
  Print #2,"  Next Ix"
  Print #2,"  Defmouse Mm$"
  Print #2,"Return"
  Print #2,""
  Print #2,"Mm_data:"
  Print #2,"' Hot spot X and Y, Number of Planes, Mask and Cursor Colors"
  Print #2,"Data ";
  For Ix=1 To 5
    Gosub X_string
    Print #2,"&";Xx$;
    If Ix<5
      Print #2,",";
    Endif
  Next Ix
  Print #2
  H1$="Data "
  H2$="&"
  Print #2,"' Mask data"
  Gosub D_write
  Print #2,"' Cursor data"
  Gosub D_write
  Close #1
  Close #2
Return
'
Procedure C_file
  Mm$=Left$(Mm$,DatposX-1)+".H"
  Open "O",#2,Mm$
  Print #2,"/* Mouse definition block */"
  Print #2
```

```
  Print #2,"MY_MOUSE ="
  Print #2,"{";Chr$(9);
  For Ix=1 To 5
    Gosub X_string
    Print #2,"0x";Xx$;",";
  Next Ix
  Print #2
  H1$=Chr$(9)
  H2$="0x"
  Print #2,"/* Mask data */"
  D_flag=0
  Gosub D_write
  Print #2,"/* Cursor data */"
  D_flag=-1
  Gosub D_write
  Print #2,"};"
  Close #1
  Close #2
Return
'
Procedure Asm_file
  Mm$=Left$(Mm$,Datpos%-1)+".S"
  Open "O",#2,Mm$
  Print #2,"*"
  Print #2,"* Mouse definition block"
  Print #2,"*"
  Print #2,"my_mouse:"
  Print #2,Chr$(9);"dc.w";Chr$(9);
  For Ix=1 To 5
    Gosub X_string
    Print #2,"$";Xx$;
    If Ix<5
      Print #2,",";
    Endif
  Next Ix
  Print #2
  H1$=Chr$(9)+"dc.w"+Chr$(9)
  H2$="$"
  Print #2,"* Mask data"
  Gosub D_write
  Print #2,"* Cursor data"
  Gosub D_write
  Close #1
  Close #2
Return
'
Procedure D_write
  For Ix=1 To 4
    Print #2,H1$;
    For Jx=1 To 4
      Gosub X_string
      Print #2,H2$;Xx$;
      If H2$="0x"
        If Not D_flag
          Print #2,",";
        Else
          If Ix<4 Or Jx<4
            Print #2,",";
          Endif
        Endif
      Else
        If Jx<4
          Print #2,",";
        Endif
      Endif
    Next Jx
    Print #2
  Next Ix
Return
'
Procedure X_string
  X$=Hex$(Inp(#1)*256+Inp(#1))
  If Len(X$)<4
    Xx$=String$(4-Len(X$),"0")+X$
  Else
    Xx$=X$
  Endif
Return
```

TUTORIAL

Step 1 - Hard Facts Pt. II

A further look under the hood.

by Maurice Molyneaux

In the previous installment of **Step 1**, I set out to give you a brief guided tour of the interiors of an ST system, discussing what various chips controlled, and which other chips they worked with. This month, rather than discussing individual parts, we're going to discuss how the system works, and why it can and cannot do various things.

Expansion

When looking at any ST but a Mega you should be aware that it is—technically—what is known as a *closed* system. No provision is made for internal expansion, i.e., adding hardware to expand the capabilities of your computer. This refers to *internal* expansion of the computer, not to adding peripheral devices to your ST (like modems, printers, digitizers, hard disks, etc.), which you obviously can do (if not, what are all those ports for?). Such closed architecture is the opposite side of the coin from *open* systems, which usually allow users to easily pop the top off of their computers and simply plug in hardware devices to do various things: add extra RAM, graphics enhancements, math coprocessors, etc. Open systems usually feature what are known as expansion *slots*. The slots are simply connectors into which the user can plug in an expansion board (or *card*). Apple II series computers (except the IIc) and most IBM PC compatibles are open systems. Computers like Atari's 520ST, 1040ST, 400, 1200XL and 65XE, Commodore's 64, 128, 16 and +4 (never heard of those last two, I'll bet), Texas Instrument's ill-fated TI-99, etc., are examples of closed architecture.

We're going to take a detour here and cover a little Atari history. As history often repeats itself, it might be wise to observe past trends and see what might be if such trends continue. If you're familiar with Atari's 8-bit computers, you might have noticed that I didn't include the 800, 600XL, 800XL and 130XE in that list of closed computers. They are not exactly open systems, but they do have some facilities for expansion. For example, the Atari 800 has four slots inside it. These were originally intended only to allow users to plug

in simple RAM expansions that would increase your original 8K or 16K of RAM to 32K or even a whopping (for that time) 48K! A few companies and individuals have produced different kinds of boards to go in these slots, but as a rule they were underutilized.

As to the 600XL and 800XL, these two machines are, in a sense, closed, except for a tiny pop-out panel on the back of the machines. Behind that panel is an edge-connector to the computer's motherboard called Parallel bus. This connector contains most of the important address and data lines of the computer and provides access to them so that users can potentially add on all kinds of devices. In fact, Atari even developed a device for this bus called the 1090XL, which was nothing more than a box that connected to the bus, and split it out into card slots—just like in an Apple II or IBM PC! The idea was to provide the ability to add RAM expansions, etc., in the box. Unfortunately, the 1090XL was a long time in coming and eventually scrapped. Without it, the expansion bus on these computers was made almost worthless (though RAM cartridges for the 600XL were produced to use this bus).

When the 130XE showed up, Atari eliminated the parallel bus connector, and replaced it with an "Enhanced Cartridge Interface" (ECI). They limited the address and data lines in this connector, but put the connector directly next to the cartridge port, through which most of the rest of the lines could be accessed if developers needed to use them.

Well, at last, a few developers began to take advantage of this expansion. Some hard disks for the 8-bits plug in through the ECI. Since the ECI connector is not the same as the XL parallel bus, usually an adapter is used to make such devices work with both XL and 130XE machines (the 65XE and XE Game System do not feature the ECI). Then, ICD produced its **MIO** device, which allows all kinds of expansion on the bus. It's what was needed when the XLs first came out. Better late than never.

Now we return to the STs. Neither the 520 or 1040 models provide any sort of parallel or processor bus. Some types of expansion are easily carried out by plugging devices into the ST's DMA (Direct Memory Access) or Cartridge (ROM) ports. To add RAM, for example, isn't as simple as plugging a card into a slot. Most methods require the user to solder wires or cables to the ST motherboard. The simplest expansions I've seen require the user to pull two chips out of their sockets (MMU and Shifter—see last issue), plug a RAM board's connectors into these sockets, and then plug the chips back into sockets on the expansion itself.

Expandability is nice, although not everyone needs it. The Mega ST units address this need—sort of. Atari hasn't given us an open system, but it has given us an internal DMA port and—a processor bus. Hmmm. It seems you can add *one* board inside a Mega by plugging into a connector which provides the full 68000 bus. It's not quite card slots; although, theoretically, an enterprising company could hook into the bus and add an expansion chassis outside the ST.

Just because Atari gave us a processor bus in the Megas doesn't mean we'll instantly be able to add an expansion box and slam cards into it.

Sound good? Maybe. Remember the XL parallel bus, 1090XL box, or the 800's internal slots? The connectors were there on those 8-bits, but they never got the kind of use they should have. Just because Atari gave us a processor bus in the Megas doesn't mean we'll instantly be able to add an expansion box and slam cards into it. It's possible that will happen, perhaps more likely than on the 8-bits, but it hasn't yet, so don't bet your bottom dollar on it. Don't run out and buy a Mega because of its "promise," because a lot of times such promises are not met.

(One quick note: At the Northeast Atari Computer Fair in October, Supra Corp. was showing an internal 20-megabyte hard disk [hard card] for the Mega STs. Please note that this device does not plug into the processor bus, but rather ties into the Megas' aforementioned internal DMA port.)

The All-In-One Micro

One thing many users fail to understand is just why various microcomputers are incompatible. Why can't you plug a C-64 disk drive into an Apple II? Run IBM software on an ST? There are myriad reasons for this, but the most common cause is that the hardware itself is completely different.

Let's take a disk drive problem as an example. Let's say you have 130XE with an Atari 1050 floppy disk drive. You also have a 520ST with its SF354 microfloppy disk drive. You want to port some data files from the XE to the ST. It would be nice if you could just plug the disk drive from one into the other and read the data. Not so easy. If, for example, you wanted to move some files from an ST disk to an XE disk, you might think to hook the SF354 drive to your XE. Doesn't sound difficult, right? Get a cable and away you go.

Wrong. Even if you could get a proper cable made, it still wouldn't work. The Atari 8-bit floppy disk drives are "intelligent," meaning they have their own ROM and disk controller systems built in. ST microfloppy drives are "dumb," and are controlled by the WD1772 disk controller inside the ST. The 8-bit doesn't have a disk controller itself, so, if the drive doesn't have its own "brains," the 8-bit can't control it.

Further, even if by some miracle you could give the 8-bit a disk controller (in software, for example), the computer's SIO (serial input/output) port doesn't transfer data at the same rates as the ST's microfloppy interface! In fact, to use an SF354 on an 8-bit would require you to build an interface board—no easy feat!

The only other way would be to buy an 8-bit drive containing a controller board that works with disk mechanisms using standard interfaces (such as the XF551), use the controller board

Hard Facts II

and swap the drive mechanism out of your ST drive. (In fact, this is being done by some dealers and hackers. They exchange the 5.25-inch drive mechanism in an XF551 with a standard 3.5-inch double-sided mechanism. Apparently this works with most kinds of 8-bit DOS, although to use both sides of the disk you'll need a DOS which supports double-sided drives—like Sparta-DOS.)

The cost and effort of building your own interface would make it prohibitive; and you'd have no better luck trying to interface an Atari 810 or 1050 drive to your ST. (And, if you did somehow manage to build the interface, you'd need a program to tell each computer just how to access the disk drive, and how to read the other computer's files!)

Such problems can occur with many types of hardware. Hard disk drives used with the ST must have a proper controller and interface. Most hard drives sold specifically for the ST have these built in, but if you want to add a "generic" hard drive, you'll have to obtain the proper controllers (these can be purchased if you know where to look).

This applies to running software too. Sometimes I think I'll scream whenever I hear people complain that "Atari should have made the ST compatible with the Amiga/IBM/Atari 8-bits/AppleII/Fred's Computer/etc." Still, it's a valid question. Why *didn't* Atari make the ST compatible with other computers?

To begin with, various computers use different parts. The microprocessor in the ST, Amiga and Macintosh is the 16-bit MC68000; in the Atari 8-bits, C-64 and Apple II family (but not the IIGS), it's the 8-bit 6502 (or a variation of it). IBM PC compatibles have Intel 8088, 8086 processors, and their more advanced models use 80286 and 80386 chips. Most CP/M machines used the 8-bit Zilog Z80. Each kind of microprocessor is different. They have different addressing modes, instruction sets, number of data and address bits, types of registers, etc. Also, chips work in "families." A 68000 is designed to work with a 68901 MFP, and the 8088 can use an 8087 math coprocessor. I suppose you could make an 8088 work with a 68901 MFP, but again, it's probably not worth the trouble.

In order to make the ST compatible with the IBM PC, Atari would have had to include an 8088 (or derivative of it) and its support chips inside the ST. They would have had to add a 6502 and support chips to make it Atari 8-bit compatible. This would not only have made the design of the computer more complex, but would have substantially raised its cost.

To complicate matters, just sticking the proper microprocessor and its support chips into the ST wouldn't do the job. For example, Atari 8-bits have a lot of custom chips inside them. Custom chips (also mentioned last time) are Integrated Circuits (ICs) designed and built for a specific computer. To make the ST fully compatible with an Atari 8-bit, we'd need not only the 6502, but the Atari 8-bit chips ANTIC, GTIA, POKEY, FREDDY, and the 8-bit ROMs. To use the 8-bit's peripheral hardware, we'd need to add the proper cartridge port, as well as an SIO port.

Atari *could* do this—since those are Atari chips. But for the ST to be Amiga compatible, Atari would need to add the custom chips from that computer (which they do not have legal rights to use), or build their own "clone" parts to do the same job. Imagine trying to make a chip which does the same job as the Amiga's Bimmer (blitter) chip, that is compatible with Amiga software, works just like in the Amiga, but doesn't infringe on the copyrights on that chip! It's next to impossible!

Even if they managed to overcome that hurdle, there's still the question of the computer's Operating System (OS). Commodore isn't going to let Atari use Intuition OS and AmigaDos, no more than Apple will license Atari to use the Mac's Finder, or the Apple II's Applesoft BASIC.

Some computers can be emulated rather easily, although dealing with formats and file compatibility can still be a problem. IBM PCs are simple to clone because IBM used "off the shelf" parts for its PCs, which anyone can buy and use, regardles of whether IBM likes it or not. Their PC-DOS OS is not in ROM, but on disk. It was designed by Microsoft and can itself be purchased under the name MS-DOS. A Mac emulator is more difficult. The Macintosh's OS is in ROM, and Apple vigorously attacks anyone who produces anything that even resembles what their ROMs do. The **Magic Sac** Mac emulator for the ST gets around this problem by providing no ROMs. You have to somehow obtain the actual Mac ROMs (either from a Mac or an agreeable Apple dealer). Then you also have to get a copy of the (copyrighted) Mac Finder software. To make the ST Mac-compatible, Atari would have had to do the same. They couldn't very well have advertised the ST as Mac-compatible and then required the user to get the Mac Finder and ROMs on his own and install them!

Oh, it can be done, one way or another. But generally speaking, it jacks the price of the machine up. Atari's marketing strategy has been "Power Without Price." To make an ST which is also Mac/Amiga/IBM/etc. compatible would make it too pricey.

I'm forgetting software emulation, like the **ST X-former**, the **CP/M emulator** or **PC Ditto**. They more or less work, but they're never as good as a hardware solution because they have to make the computer *pretend* it's something it isn't. Since the commands for an 8088 processor are different than those for a 68000, for the ST to be able to run MS-DOS (IBM) software, the emulator must translate the 8088 instructions into a form understandable by the 68000. Then the 68000 must do its job and translate its data back to a form understood by the program being run. This is usually *slow*. Most software emulators run between 30% and 70% as fast as the real thing. But, I hasten to add, they do run. They may not be 100% compatible with, or as fast as the real thing, but if the program you need to run works with a software emulator, then it's probably worth having (and it's usually much cheaper to obtain a software emulator than a hardware one.)

Mission: POSSIBLE

I suppose now that I've listed all these things your ST can't do, I should balance the bad news with good by discussing some things it can. Let's start by looking at expandability. True, the ST doesn't have card slots, and there isn't (as I write this) anything that uses the Mega's processor bus, but you can add some kinds of hardware to your ST.

The most common hardware upgrade is the RAM expansion. Most often these are performed on 520STs to give them the same total RAM as a 1040ST: 1 megabyte (1024K). There are also upgrades for 1040STs, as well as boards which will push either ST into the 2- to 4-megabyte range. Most of these large expansions feature a board to

which the user must add RAM chips, usually of the (somewhat expensive) 1-megabit variety. The chips for such large upgrades must usually be purchased separately of the board.

Most common RAM expansions are of a set RAM size and all necessary chips are provided. The boards usually work by hooking into the address lines of the MMU and Shifter chips. A few (as mentioned earlier) can be plugged in, but most require soldering jumper wires, cables, or for you to move or replace some components. Some might even require you to "piggyback" components (placing one chip on top of another and soldering the top one's legs to those below), which is delicate and sometime dangerous work.

A far less common upgrade (now) is to put TOS in ROM. If you purchased a 520ST prior to January 1986, it probably didn't have TOS in ROM (in such a case you need a TOS system disk to boot the machine). Using a disk-based TOS takes away about 200K of RAM from your use. If you have an older 520ST and still must boot TOS from a disk, you should get the ROM TOS—*now*! Installing it is a simple operation, merely requiring the technician to remove the two 16K boot ROMs and insert the TOS chips into the waiting sockets. The only loss is the colorful boot screen (I always liked those color-cycling Fuji logos), which the gains far outweigh.

A battery backup clock is another common upgrade. When files are saved the current time/date settings are stamped on the file. The time/date is set from the Control Panel accessory, which you may not always have present on the Desktop. It's also easy to forget to set the clock when you boot up. A clock upgrade adds a battery backup unit to your ST (the board, because of the battery, holds the correct time even when the computer has been turned off). When the computer is booted, the current time/date from the clock are fed to the system's own real-time clock, insuring the correct date and time are always set.

Such upgrades come in a number of forms. Some are cartridges, plugging in externally (some of these extend the cartridge port, permitting you to plug another cartridge in even with the clock in the cartridge port—though I've heard of problems with this method). Others are small units which are interfaced into the IKBD chip socket on the bottom of the keyboard, or plug into one of the ROM chip sockets. The internal kind tend to be better because you don't always end up having a cartridge hanging out of the side of your ST. The Mega STs have a battery backed-up clock as part of their standard equipment.

When most people think of cartridge ports, they generally think of video games. Such ports can be used for all kinds of input, not just plugging in **PacMan**. That the ST has a cartridge port is nice, and something that should not be overlooked. Without it, imagine how difficult it would be to implement something like the Magic Sac! The cartridge port allows such devices to be plugged right in.

As history often repeats itself, it might be wise to observe past trends and see what might be if such trends continue.

Sure, it's not the same as having six or eight card slots, but it's something Atari could easily have left out. With it, I can plug in a digitizer, a cartridge containing numerous desk accessories, the Magic Sac, StereoTek 3-D glasses, or even a system diagnostic unit (used by Atari service centers to check machines for defects), without having to unplug other devices (my printer, modem, hard disk, etc.).

My only advice: *Don't touch the cartridge port!* If at all possible avoid any contact with it, particularly when the machine is on. The port exposes many critical address lines on the motherboard, and you don't want to risk building up static electricity, or causing a short, in such a delicate area.

Ciao!

That's Italian for farewell (and, no, it's not spelled chow), but only with people you really know, otherwise it's *arrivederci*. And, since we've become familiar over the past months, I'll close with it. Tune in next time when we'll be getting away from hardware-related topics (at last).

Addenda

Again, more comments and corrections on previous **Step 1s**.

—In **Processed Prose** (issue #17), I was discussing **ST Writer** version 1.70 and **1st Word** version 1.06. I forgot to mention that while versions of 1st Word were freely distributed with some ST systems, that bundling was discontinued some time ago. 1st Word can be purchased from Atari Corp., but it is no longer a "freebie." I do not know what version number the latest for-sale version is.

As to ST Writer, since I wrote **Processed Prose**, a significant upgrade has taken place. As I write this, ST Writer version 2.3 is now available (public domain). This version works just like 1.70 and 1.80, but adds some GEM goodies. You can activate the mouse and use a special GEM menu bar in place of the usual command menu screen, or even move your cursor or scroll pages using the mouse. However, the mouse can easily be disabled, allowing the program to remain much quicker and nimbler than 1st Word. Contact your local users group or peruse some BBS systems if you wish to obtain the latest ST Writer upgrades. They can also be found in ANALOG's Atari SIG on Delphi.

Allergic to all things Commodore, Maurice Molyneaux is an author and artist who—when not writing articles for ST-Log*—continues to struggle with a recalcitrant 8-year-old science fiction novel, paints, illustrates, and uses his ST for "every conceivable task." Despite a ridiculously French name, he claims to having been born in Vicenza, Italy, and denies vicious rumors that he eats escargots and calamari when computing. His DELPHI username is MAURICEM.*

Decimal Destr GAME

by Kevin Kennedy

Low Resolution

Some time in the future, thousands of asteroids were detected floating toward Earth. Fortunately, because they were found soon enough, hundreds of robot ships were sent to destroy them. These ships were controlled by people on Earth. Though they had only a few controls, including rotate, thrust and fire, they were able to destroy most of the asteroids, but there were still some heading for Earth. They had anticipated this problem and built a sophisticated and expensive defence system. To our surprise, the guidance module was never installed. Once again, people were called to help save Earth.

The method of operation for these systems was completely different. The Earth was divided into ranges. The length of the ranges depended on the expected amount of asteroids. The range would be smaller if more asteroids were expected in that area and larger if there were only a few asteroids expected. The ranges were measured in miles. Because

computers were supposed to be handling this system, some ranges were specified in fractions of miles. For example, in an area where many asteriods would fall the range would be small, like .25 miles. Missiles were used to blow these asteroids out of the sky, and were launched by specifying their location in the range. For example, if a missile were to be launched in the middle of a one mile range, the value .5 would be entered into the computer.

In Decimal Destroyer, you have been called to be one of the defenders of Earth. Before the game starts, you will be asked what level you want to start on. Levels go up to seven, one being the easiest. When the game starts, you must enter a number between the ranges. The range extends from the number on the left of the window, always zero, to the number on the right. You can enter any number between 0.000 to 9.999, not going past the thousandth place. You will be told if your number is out of the ranges.

During the game, asteroids are descending and must be hit before they reach the ground. You can miss an asteroid three times before you will be told what number you should have entered.

When you destroy either two or four asteroids, depending on what level you are on, you advance a level. You are allowed five misses per game. A miss occurs when an asteroid hits the ground. The game ends when you either run out of misses or you complete level seven. You will then be rated from excellent to try again.

The Program

Decimal Destroyer is written in Personal Pascal by OSS and CCD. It uses GEM calls so it needs all the Pascal GEM support files and must be compiled with GEM selected.

Decimal Destroyer uses XBIOS calls for sound, random number generation, and to shake the screen up and down. The shaking effect is done by quickly adding and subtracting 1280 to the physical screen base.

Because pascal does not have any raster copy functions, I used an old animation technique. Rather than drawing, than erasing the entire object, only the parts of the object that need to be drawn or erased are handled. For example, on the rocket only the nose and the tips of the fins are drawn and only the bottom part is erased. When the rocket ascends it appears to be a solid object. The problem with this method is that objects can travel only in one direction and only one step at a time.

One of **Mr. Kevin Kennedy's** *projects in a Colorado State Science Fair, Computers for the Impaired, won an award from the IEEE (Institute of Electrical and Electronic Engineers).*

```
PROGRAM Decimal_Destroyer;

{**************************************
 ****      DECIMAL DESTROYER        ****
 ****   Written by Kevin Kennedy    ****
 ****         using                 ****
 ****      Personal Pascal          ****
 ****           by                  ****
 ****        O.S.S.                 ****
 **************************************}



CONST
 {$I GEMCONST.PAS}

TYPE
 converted = STRING [5];
 {$I GEMTYPE.PAS}

VAR
 level,ml,dia : INTEGER;
 range : REAL;

{$I GEMSUBS.PAS}

{**** O.S. FUNCTION CALLS ****}

FUNCTION physbase : LONG_INTEGER;
 XBIOS(2);

PROCEDURE setscreen(a,b : LONG_INTEGER ; c : INTEGER);
 XBIOS(5);

FUNCTION sound_chip(data,register : INTEGER) : INTEGER;
 XBIOS(28);

FUNCTION random : LONG_INTEGER;
 XBIOS(17);

{**** SOUND PROCEDURE ****}

PROCEDURE sound(tone : LONG_INTEGER ; volume,register,noise:INTEGER);
 VAR
  dummy,msb,lsb,enable,toneh,tonel,sel,ensve : INTEGER;
 BEGIN
  ensve := sound_chip(ensve,7);
  toneh := INT(tone DIV 256);
  tonel := INT(tone MOD 256);
  lsb := ensve & 64;
  msb := ensve & 128;
  CASE register OF
   0 : sel := 62 ;
   3 : sel := 55 ;
  END;
  enable := lsb+msb+sel;
  dummy := sound_chip(enable,7+128);
  dummy := sound_chip(volume,8+128);
  IF register<3 THEN
   BEGIN
    dummy := sound_chip(tonel,0+register+128);
    dummy := sound_chip(toneh,1+register+128);
   END
  ELSE
   dummy := sound_chip(noise,6+128);
 END;

{**** MISC. FUNCTIONS & PROCEDURES ****}

PROCEDURE conv(in1 : REAL ; VAR out : converted) ;
```

```
  VAR
   cnv : INTEGER;
  BEGIN
   FOR cnv := 1 TO 4 DO
    BEGIN
     out[cnv] := Chr(Trunc(in1)+48);
     in1 := (in1-Trunc(in1)) * 10;
    END;
  END;


 FUNCTION establish_difficulty(level : INTEGER) : INTEGER;
  VAR
   temp : INTEGER;
  BEGIN
   CASE
    level OF
     1:range := 1;
     2,3:range := (INT(random &    3) +   1) * 0.5;
     4,5:range := (INT(random &    7) +   1) * 0.25;
     6,7:range := (INT(random & 1023) + 500) * 0.001;
    END;
   CASE
    level OF
     2,4,6 : establish_difficulty := 2;
     1,3,5,7 : establish_difficulty := 4;
    END;
  END;


 PROCEDURE show_tally(number,hits : INTEGER);
  VAR
   dialog1,dialog2,dialog3 : STRING;
   dialog4 : STR255;
   p : REAL;
   x : INTEGER;

  BEGIN
   dialog1 := '[1][       GAME OVER|Out of __ you destroyed __.|';
   p := hits / number * 100;
   dialog2 := '                TRY AGAIN';
   IF p > 89 THEN dialog2 := '       EXCELLENT';
   IF (p > 79) AND (p < 90) THEN dialog2 := '          VERY GOOD';
   IF (p > 59) AND (p < 80) THEN dialog2 := '            FINE';
   IF (p > 40) AND (p < 60) THEN dialog2 := '            SO-SO';
   x := number DIV 10;
   dialog1[29] := CHR(number DIV 10 + 48);
   dialog1[30] := CHR(number MOD 10 + 48);
   dialog1[46] := CHR(hits DIV 10 + 48);
   dialog1[47] := CHR(hits MOD 10 + 48);
   dialog3 := '][ OK ]';
   dialog4 := CONCAT(dialog1,dialog2,dialog3);
   dia := Do_Alert(dialog4,1);
  END;

(**** ANIMATION PROCEDURES ****)

 PROCEDURE clip;
  BEGIN
   Set_Clip(60,1,289,148);
  END;

 PROCEDURE unclip;
  BEGIN
   Set_Clip(0,0,639,200);
  END;


 PROCEDURE draw_rocket(x,y : INTEGER);
  BEGIN
   clip;
   Line_Color(2);
   Line(x+2,y+2,x+4,y);
   Line_To(x+6,y+2);
   Line(x,y+11,x+1,y+10);
   Line(x+7,y+10,x+8,y+11);
   Line_Color(0);
   Line(x,y+16,x+2,y+14);
```

```
    Line_To(x+6,y+14);
    Line_To(x+8,y+16);
    unclip;
  END;

 PROCEDURE draw_asteroid(x,y : INTEGER);
  BEGIN
    clip;
    Line_Color(6);
    Line(x,y+10,x+2,y+12);
    Line_To(x+3,y+11);
    Line_To(x+5,y+13);
    Line_To(x+7,y+11);
    Line_To(x+8,y+12);
    Line_To(x+10,y+10);
    Plot(x,y+4);
    Plot(x,y+7);
    Plot(x+11,y+8);
    Line_Color(0);
    Plot(x,y+9);
    Plot(x,y+5);
    Line(x,y+3,x+1,y+2);
    Line_To(x+2,y+2);
    Line_To(x+4,y);
    Line_To(x+6,y+2);
    Line_To(x+7,y+2);
    Line_To(x+8,y+1);
    Line_To(x+9,y+1);
    Line_To(x+11,y+3);
    unclip;
  END;

(**** Graphics procedures ****)

 PROCEDURE colors;
  BEGIN
    Set_Color(0,0,0,1000);
    Set_Color(1,0,0,0);
    Set_Color(2,1000,0,0);
    Set_Color(3,1000,0,0);
    Set_Color(6,0,1000,1000)
  END;

 PROCEDURE draw_screen;
  VAR
    out,r : converted;
    cpy : INTEGER;
  BEGIN
    unclip;
    colors;
    Clear_Screen;
    Line_Color(1);
    Paint_Color(3);
    Frame_Round_Rect(49,8,230,150);
    Line(64,148,64,148);
    Line(164,145,164,148);
    Line(264,148,264,148);
    Text_Color(2);
    Draw_String(160,160,'Misses Left:');
    r := '';
    r := Chr(ml+48);
    Draw_String(272,160,r);
    Draw_String(40,140,'0');
    r := '';
    conv(range,r);
    Draw_String(340,140,r);
    out := '.....';
    out[1] := r[1];
    FOR cpy := 2 TO 4 DO
      out[cpy+1] := r[cpy];
    Draw_String(279,140,out);
    Draw_String(30,160,'Level:');
    Draw_String(78,160,CHR(level+48));
  END;

 PROCEDURE rumble;
  VAR
```

```
  lloop : INTEGER;
  pb : LONG_INTEGER;
 BEGIN
  pb := physbase;
  FOR lloop := 1 TO 1000 DO
   BEGIN
    IF Odd(lloop)
      THEN setscreen(-1,pb-1280,-1)
      ELSE setscreen(-1,pb,-1);
    sound(0,15-lloop DIV 57,3,lloop);
   END;
  sound(0,0,3,0);
  setscreen(-1,pb,-1);
 END;


FUNCTION input(rng : REAL) : REAL ;
 LABEL 2,3;
 VAR
  gb : DIALOG_PTR;
  ln1,ln2,dui,conv,te : INTEGER;
  txi : STR255;
  input2 : REAL;
 BEGIN
  gb := New_Dialog(3,2,21,35,4);
  ln1 := Add_DItem(gb,G_Text,0,0,1,33,1,0,401);
  Set_DText(gb,ln1,'Enter a number within the ranges',3,TE_Center);
  ln2 := Add_DItem(gb,G_FText,15,14,2,5,1,0,401);
  Set_DEdit(gb,ln2,'.___','9999','',3,TE_Center);
  2 : dui := Do_Dialog(gb,ln2);
  Get_DEdit(gb,ln2,txi);


  FOR conv := 1 TO 4 DO
   BEGIN
    te := Ord(txi[conv])-48;
    IF (te<0) OR (te>9) THEN txi[conv] := '0';
   END;
  input2 := 0;
  input2 := (Ord(txi[1])-48)+((Ord(txi[2])-48)* 0.1)+((Ord(txi[3])-48)* 0.01)+
            ((Ord(txi[4])-48)* 0.001);
  input := input2;
  IF (input2 >-0.999) AND (input2 < rng+0.001) THEN GOTO 3;
  dia := Do_Alert('[1][ Number is|out of range][ OK ]',1);
  GOTO 2;
  3 : End_Dialog(gb);Delete_Dialog(gb);
  Paint_Color(0);
  Paint_Rect(0,165,319,35);
 END;

PROCEDURE explode(xpos,ypos : INTEGER);
 VAR
  radius,r,g,b : INTEGER;
 PROCEDURE oval;
  BEGIN
   r := Int(random & 1000);
   g := Int(random & 1000);
   b := Int(random & 1000);
   Frame_Oval(xpos,ypos,radius,radius);
   Set_Color(1,r,g,b);
   sound(0,10,3,Int(random & 31));
  END;
 BEGIN
  clip;
  Line_Color(1);
  FOR radius := 0 TO 30 DO
   BEGIN
    oval;
   END;
  Line_Color(0);
  FOR radius := 30 DOWNTO 0 DO
   BEGIN
    oval;
   END;
  sound(0,0,3,0);
  unclip;
  draw_screen;
 END;
```

```
PROCEDURE move_asteroid(xpos,ypos,ydest : INTEGER);
 VAR
  ay : INTEGER;
 BEGIN
  FOR ay := ypos TO ydest DO
   BEGIN
    draw_asteroid(xpos,ay);
    sound(ay,10,0,0);
   END;
  sound(0,0,0,0);
 END;

PROCEDURE launch(xpos,ydest : INTEGER);
 VAR
  ay : INTEGER;
 BEGIN
  FOR ay := 160 DOWNTO ydest DO
   BEGIN
    draw_rocket(xpos,ay);
    sound(0,10,3,ay DIV 5*Ord(ay>0));
   END;
  sound(0,0,3,0);
 END;

{**********************************************************************}
{**********************     main game     **********************}
{**********************************************************************}

 PROCEDURE do_game(active : INTEGER ; VAR h : INTEGER);
  VAR
   asteroid_ary : ARRAY [1..4,1..3] OF INTEGER;
   drop,gme,shoot,ax,hit : INTEGER;
   num : REAL;

  PROCEDURE setpars(asteroid : INTEGER);
   BEGIN
   asteroid_ary[asteroid,1] := 1;
   asteroid_ary[asteroid,2] := Round(Int(random & 255) * 0.784)+61;
   asteroid_ary[asteroid,3] := -10;
  END;

  PROCEDURE smash(asteroid : INTEGER);
   VAR
    move,astx,asty: INTEGER;
    alert_txt : str255;
    alert2 : converted;
    astnum : REAL;
   BEGIN
    astx := asteroid_ary[asteroid,2];
    asty := asteroid_ary[asteroid,3];
    move_asteroid(astx,asty,143);
    rumble;
    astnum := (astx-61)*range/200;
    alert_txt :=
    '[1][You missed. You|should have tried|around _.___][ RETURN ]';
    conv(astnum,alert2);
    alert_txt[46] := alert2[1];
    FOR move := 2 TO 4 DO
     alert_txt[46 + move] := alert2[move];
    dia := Do_Alert(alert_txt,1);
    asteroid_ary[asteroid,1] := 0;
   END;

  PROCEDURE clear_ary;
   VAR
    er,er2 : INTEGER;
   BEGIN
    FOR er := 1 TO 4 DO
      FOR er2 := 1 TO 3 DO asteroid_ary[er,er2] := 0;
    drop := 0;
   END;

  PROCEDURE ads(asteroid : INTEGER);
   BEGIN
    asteroid_ary[asteroid,3] := asteroid_ary[asteroid,3] + 37;
    move_asteroid(asteroid_ary[asteroid,2],asteroid_ary[asteroid,3]-37,
    asteroid_ary[asteroid,3]);
   END;
```

```
PROCEDURE advance;
 VAR
  asteroid : INTEGER;
 BEGIN
  FOR asteroid := 1 TO 4 DO
   IF asteroid_ary[asteroid,1] <> 0 THEN ads(asteroid);
 END;

FUNCTION check_hit(msx : INTEGER) : INTEGER;
 VAR
  asteroid,hit,asx : INTEGER;
 BEGIN
  hit := 0;
  FOR asteroid := 1 TO 4 DO
   BEGIN
    asx := asteroid_ary[asteroid,2];
    IF (asteroid_ary[asteroid,1] <> 0) AND (msx+4 < asx+12)
    AND (msx+4 > asx) AND (hit = 0) THEN
     hit := asteroid;
   END;
  check_hit := hit;
 END;

PROCEDURE blowup(asteroid,msx : INTEGER);
 BEGIN
  hit := hit + 1;
  launch(msx,asteroid_ary[asteroid,3]+6);
  explode(asteroid_ary[asteroid,2]+6,asteroid_ary[asteroid,3]+6 );
  asteroid_ary[asteroid,1]:= 0;
 END;

PROCEDURE miss(msx : INTEGER);
 BEGIN
  launch(msx,-20);
 END;

FUNCTION check_miss : INTEGER;
 VAR
  asteroid,astmp : INTEGER;
 BEGIN
  FOR asteroid := 1 TO 4 DO
   BEGIN
    IF (asteroid_ary[asteroid,1] <> 0)
    AND (asteroid_ary[asteroid,3] > 130) THEN
     check_miss := 1;
   END;
 END;

FUNCTION ground_hit : INTEGER;
 VAR
  asteroid,hit : INTEGER;
 BEGIN
  hit := 0;
  asteroid := 0;
  WHILE (hit = 0) AND (asteroid < 4) DO
   BEGIN
    asteroid := asteroid + 1;
    IF (asteroid_ary[asteroid,3] > 180)
    AND (asteroid_ary[asteroid,1] <>0) THEN hit := asteroid;
   END;
  ground_hit := hit;
 END;



FUNCTION ast_left : INTEGER;
 VAR
  asteroid,lft : INTEGER;
 BEGIN
  lft := 0;
  FOR asteroid := 1 TO 4 DO
   BEGIN
    IF asteroid_ary[asteroid,1] <> 0 THEN
     lft := lft + 1;
   END;
  ast_left := lft;
 END;
```

```
 BEGIN
  clear_ary;
  hit := 0;
  draw_screen;
  LOOP
   drop := drop + 1;
   IF drop < active + 1 THEN
    setpars(drop);
   advance;
   EXIT IF (ast_left < 1) OR (ml < 1) ;
   num := input(range);
   ax := Round(num*200/range)+60;
   shoot := check_hit(ax);
   IF shoot <> 0
    THEN
     blowup(shoot,ax)
    ELSE
     miss(ax);
   IF ground_hit <> 0 THEN
    BEGIN
     smash(ground_hit);
     ml := ml -1;
    END;
   draw_screen;
  END;
  h := hit;
 END;


PROCEDURE main; (main control loop)
 VAR
  hit : BOOLEAN;
  hits,astnum,totalh,totasts : INTEGER;
 BEGIN
  totalh := 0;
  totasts := 0;
  draw_screen;
  ml := 5;
  dia := Do_Alert(
  '[2][What level do you|want to start on?|   1=easy][ 1 | 2 | 4 ]',1);
  level := TRUNC(dia * 1.49);
  LOOP
   astnum := establish_difficulty(level);
   do_game(astnum,hits);
   totasts := totasts + astnum;
   totalh := totalh + hits;
   EXIT IF (ml < 1) OR (level > 6);
   level := level + 1;
  END;



  show_tally(totasts,totalh);
 END;


BEGIN
 IF Init_Gem >= 0 THEN
  BEGIN
   dia := Do_Alert(
  '[1][Decimal Destroyer by|Kevin Kennedy using|Personal Pascal by OSS][ OK ]'
   ,1);
   Hide_Mouse;
   Clear_Screen;
   draw_screen;
   REPEAT
    main;
    dia := Do_Alert('[2][Do you want to play again?][ YES | NO ]',1);
   UNTIL dia = 2;
   Set_Color(1,1000,1000,1000);
   Set_Color(0,0,0,0);
    Set_Color(3,0,0,700);
   Show_Mouse;
   Exit_Gem ;
  END
END.
```

IAN'S QUEST

by Ian Chadwick

"Whatever became of. . . ."

A lot of conversations begin with these words when people learn of my past association with Batteries Included. People want to know what happened. More often, people want to know what happend to the software BI had announced that hasn't yet seen the light of day—the Elite series in particular. Somehow, in the acquisition of BI by Electronic Arts, a lot of products appeared to fall by the wayside.

A lot of the time I have to answer, "I dunno," because EA never let me in on their plans. When I left BI, among the projects in the works for the ST were **PaperClip Elite, Contact** (aka Bute, the terminal emulator, once called Termulator until someone realized the name was already in use), the **Thundersaurus**, **B/Graph Elite**, **Consultant Elite**, a revised **Thunder** and a few other programs. As far as I can tell, none of these are going to be released by EA, at least within the Age of Mammals.

That's too bad. There were a lot of good products under development back before New Year's 1987. Top of the list and the closest to completion was Scott Northmore's Consulant Elite.

Scott had originally written an impressive database for the MS-DOS market, called Genapp. It was one of those products up in the application stratosphere—complex, powerful and flexible. He wrestled the user interface under a GEM shell, which made it considerably easier to use and more approachable than the command-based ogres, dBase and Rbase, but sharing their capabilities nonetheless.

Once running under GEM, the next phase was to move it over to the ST. But DRI's GEM in the PC/MS-DOS environment isn't 100% the same as GEM in an ST, and translations aren't quite as simple as one might expect. Still, Scott managed to tame the tiger to the point where we could honestly say it was 90% finished and ready for outside testing. The prototype worked, and the database engine itself performed properly—it was mostly a matter of tightening the user interface. It also promised to be the most powerful database manager available for the ST by far. Then the troubles began.

Michael Reichmann, president of BI, left for what appeared to be greener pastures (a company called Laser Friendly, involved in desktop publishing—soon to become a haven for several ex-BI employees). With him went a lot of the drive and determination that kept BI going. The remaining management was indecisive and insecure. There were a lot of meetings, and thousands of words were spoken, but software publication—not to mention development and testing—virtually came to a standstill while psychic efforts were expended in useless hair-pulling and tooth-gnashing over the financial crunch BI found itself in. Those of us who wanted to continue on were powerless to get the wheels turning again. We marched towards the inevitable: the sale of the company.

Do the designers think we live in a swamp? Think there might be a little bit of prejudice in the making of this product?

No one told Scott or any of the other software authors about this however. Management dithered and hesitated and hemmed and hawed while he worked diligently on his own. We were forbidden to discuss the situation with any of the authors in case it jeopardized the negotiations with Epyx and later EA. That only lead to a lot of angry, frustrated programmers and developers. Not to mention the angst many of us at BI felt over this sort of treatment.

Suddenly, or so it seemed, the whole thing ended. BI went into receivership, people were fired without warning—most without being given proper notice or sufficient separation pay (myself included). But it was people like Scott who were really left with the short end of the stick.

Scott had brought Consultant Elite to the beta test stage, but what now? EA held the rights to all products, even those in development, and were slow to release them—even those they had no intention to publish. Scott finally got the rights for his program back in December—nine months or so after the sale. For nine months he couldn't legally do anything with Consultant. All that work didn't generate any income.

Scott realized they would have to return the rights sooner or later, so he was resourceful enough to approach several other ST software publishers in the meantime. But with the rights in limbo for so long, one could or would make a commitment to the program. Worse, perhaps, several companies were interested but realized the responsibility for supporting such a high-end product was a major undertaking outside their realm. They were impressed and excited by the program, but lacked either resources or stamina to provide the requisite support staff Consultant required. And that bodes ill for the development of serious ST software in the future.

So Consultant Elite is still unpublished—potentially one of the most powerful database programs for the ST, a professional-level product that won't see the light of day because Scott can't find a publisher willing to support it. That's not to say there aren't good database programs out there now, but the likelihood of seeing more of such or better programs in the future is slim (competition means better end products and real choice—and very few publishers seem willing to enter the fray).

A similar fate befell B/Graph Elite. It seems no one can see the need for a serious statistical/graphing program (with a spreadsheet data-entry interface) in the ST market. Sure, these programs

sell well, and are in demand in the PC/MS-DOS world, but ST software publishers seem to shy away from the upper-end products. Either that or they haven't the foggiest idea what they should be selling.

All this makes me somewhat nervous. It's like we're doomed to seeing the ST become a great game machine, or a super midi-controller, but with a dearth of products aimed at the business/professional user. And face it folks, professional users are staying away in droves.

After all, what do you have for the ST we can truly say falls into the professional category? Anything the likes of 1-2-3 (or Quattro or Excel), Sidekick, Desqview, Symphony, Framework, Paradox, Ready, Keyworks, Ventura, Grammatik, Prolog, Tornado Notes, and so on? I doubt that any of their publishers will release for the ST. It's seen as a game machine by most outsiders. And in order to change that attitude, we have to see more top-level products produced and sold by existing ST publishers.

Sure, Word Perfect Corp. released **Word Perfect 4.1**—so far the only publisher of note in the PC/MS-DOS world to do anything of that sort. So, at last, we have a word processor that can be considered professional quality. What else? The sad truth is that we simply don't have much (if any) software in the PC/MS-DOS league.

I really don't want to pick holes in every product and whine about what we don't have. In truth, a lot of what's out there is pretty good, but it's all based on a perception of home market needs and prejudices; it's pretty much still a cottage industry. Maybe we're to blame. Maybe most ST buyers don't want the business and professional-level software; they want lightweight products, simple utilities and lots of games. In that case, Word Perfect may well fail to sell big in the ST market, despite the quantity of features it offers (it towers over any other ST word processor).

Of course, the home market is where Atari seems to be aiming its best shot. I don't see a single ST in use in any corporate environment. Can anyone name a member of the Fortune 1000 that uses STs? Apple has managed to get the Mac into a lot of businesses. Why not Atari?

Well, in part, it's because they can't approach a major business seriously without some solid software to offer them. Can you imagine Atari trying to sell to a major corporation?

"Well, no, we don't have networking, no simple means to move files from your existing PCs, no suitable terminal emulation for your mainframe connection, software lacks full compatibility with your 1-2-3 and dBase files, and the keyboard is mediocore, but you can play some great games on it. . . ."

Maybe I'm just in a bad mood. I get depressed when I go into a software shop and see the shelves full of PC/MS-DOS products. I'd like to see things change and find products like Scott's Consultant Elite (or whatever he's calling it now—maybe Genapp Elite?) and B/Graph Elite in my local software store soon, before the ST becomes the C64 of the '80s.

Maybe I'm just in a bad mood. I get depressed when I go into a software shop and see the shelves full of PC/MS-DOS products.

Ah, so that's why they never got in touch.

Remember back when, I made a fuss about Sublogic not doing a **Flight Simulator** scenery disk for my own area (Toronto)? I even wrote them *twice* and begged them to release the information for creating a scenery disk database so local users could design their own places to fly (or create fantasy places—why not?). In response, I got a written equivalent of the cold shoulder from them.

Well, imagine my surprise when I discovered that Scenery Disk 11—the "Detroit/Lake Huron" disk—has Toronto on it! And a good deal of southern Ontario too. But they don't mention it anywhere on the package. I booted my FS2 and skewed over to find out why.

If you know the city, you'll understand why Sublogic is reluctant to admit to having Toronto on any their disks. Toronto is, after all, not an American city. Just because it's the largest city in Canada, with roughly five million people in the combined metro and suburban regions, doesn't mean they should give it any special treatment, right? And the fact there are more ST owners in Soutern Ontario than anywhere else in the country shouldn't bother them either, I suppose.

Well, imagine my disappointment: Toronto is given one single building—the CN Tower—around which we can fly. Detroit—roughly the same size and population—has oodles of interesting places to see. At night, FS2's Toronto is as black, bleak and dreary as a morgue. And to add insult to injury, Pearson International Airport—as busy as Chicago's O'Hare—is missing from the booklet and map listing and was given no buildings in the database, not even a lousy hangar! And where are the dozens of large satellite communities that surround this city? Consigned to FS2 limbo, no doubt. Finally, what's with all the lakes that look like half-cooked fracted pancakes that cover the northern landscape? Do the designers think we live in a swamp?

Think there might be a little bit of prejudice in the making of this product? Look, as a Canadian, I'm accustomed to getting the short shrift from Americans who think that the world ends at the border, but this is carrying it too far; it's a smack in our faces. It would have been better had Sublogic not included Canada at all than to insult us with such a piddling poor effort.

My love of FS2 has suffered a serious blow with this scenery disk. And I'm not the only one who feels this way up here. Fellow enthusiasts have discussed it with me in several stores—all equally grieved at the offhand manner in which our own area was handled. It casts a dark shadow on any claims to accuracy that the scenery disks might ever carry in the future. Sublogic, are you listening? If you ever want us on your side again, you'd better recall those disks and fix them up, soon, before the damage is irreversible.

STPlus•STPlus•STPlus•STPlus

P.O. 1197, Berkeley, Ca. 94701

We all want the ST to grow, so let's BUY MORE SOFTWARE and discourage pirating!

BUSINESS

DBMan 4.0	175.00
Datamanager	56.00
Superbase	104.95
Trimbase	69.95
Phasar	63.95
Zoomracks 2	84.95
Base 2	42.95
The Informer	69.95
Wordperfect	189.95
1st Word Plus	69.95
WordUp!	74.95
Best Accounting	279.95
Equal Plus	139.95
Inventory Mgr.	69.95
Rolobase Plus	63.95
Logistix Spread	104.95
Microlawyer	49.95
Payroll Master	69.95
Construction EST.	35.00
Microsoft Write	94.95
Datatrieve	35.00
STOneWrite	48.95
VIP GEM	104.95
DacEasy Payroll	48.00
DacEasy Acctg	52.00
WordWriter ST	56.00
SwiftCalc	56.00
EZ Calc by Royal	48.95
Analyze Spread	104.95
Final Word	99.95
PublishingPartner	140.00
T-works Publish It	104.95
EZData Base	48.95
Chart Pak	35.00
Compute Roots	27.95
Thunder NEW!	28.95
Habawriter 2	48.95
Text Pro	35.00
Becker Text	62.95
Expert Opinion AI	59.95
Time Link	35.00
Partner ST	48.95
Labelmaster Elite	35.00
ST Accounts	149.00
The Juggler	35.00
Max Pack	35.00
Stuff	27.95
Flash 1.5	21.00
SBT accting ea.	275.00
Omni Res	27.95
Turbo ST(~blitter)	35.00
Dollars & Sense	69.95

GRAPHICS

Degas elite	55.95
CAD 3D 2.0	63.95
Cyber Paint	49.00
Quantum 4039	
Adv Art Studio	26.00
Spectrum 512	49.00
EzDraw&Superch	104.95
Canon Scanner	1040.0
GFA Artist 1000cl	55.95
Drafix 1	139.95
Athena 2	69.95

GAMES

Gunship	35.00
Shadowgate	35.00
Uninvited	35.00
Mouse Quest	14.00
Slaygon	27.95
Barbarian	27.95
Obliterator	27.95
Guantlet	35.00
Dark Castle	27.95
F-15 Strike Eagle	27.95
Star Trek-Rebel U.	27.95

MUSIC

Passport	
Master Tracks	280.00
MasterTracks Jr.	104.95
Midisoft Studio	69.95
Hybrid Arts	
Smpte Track	499.95
Sync Track	299.95
EZ Track Plus	48.95
Midiscore	call
EZ Score Plus	104.95
DX-Android	139.95
CZ-Android	69.95
Gen-Patch	104.95
D-50 Editor	call
Voice Masters	
Yamaha TX81Z	69.95
Roland AJ 1 & 2	69.95
Yamha 21,27,100	55.95
Oasis Editor	175.95
Hybridswitch	21.95
ADAP Smptecue	175.95
upgrade old box	70.00
MIDI-MAZE	27.95
ADAP 2 direct to 60mghd sampler	2795.0
Midiplexer	249.95
Dr. T's	
KCSequencer	199.95
KCS 1.6 w/ PVG	289.95
MIDI rec studio	27.95
NEW Copyist	139.95
Cheetah Midi Inst.	
MK5 Keyboard	249.95
MK5 II	399.95
MK5 V	549.95
MK7 VA	794.95
MIDI Drum	339.95
Power Play Drum	369.95
Drum Interfacer	119.95
Synth Module	534.95
SMPTE to MIDI	349.95
DX Heaven edltr	104.95
Korg, Kawai,etc	call
CZ Patch editor	79.95
CZ patches	39.95
DX patches	39.95

How would you like to be an ST dealer? If you are interested, I am looking for a few limited partners to work with in areas which lack ST support. This is not a solicitation by Atari or to circumvent Atari's network but an invitation to work with an established dealer to set up new dealerships. I am especially interested in VAR's for business and education.

PROGRAMMING

GFA Basic	48.95
GFA Book	35.00
GFA Compiler	42.00
Mark Williams 'C'	125.00
Laser 'C'	159.95
Cambridge Lisp	139.95
RAID	27.95
Fast Editor	35.00
Alice Pascal	69.95
OSS Pascal	59.95
Fortran 77 GEM	139.95
BCPL	104.95
Modula2 dev. kit	104.95
Assempro	48.95
Fast Basic	56.95
True Basic	69.95

EDUCATIONAL

Arakis Series	14.00
Unicom Series	27-35
True Basic Stuff	69.95p

GAMES

Tanglewood	27.95
Test Drive	35.00
Chessmastr2000	32.95
StarGliderbw&cl	32.95
Hunt for Red Oct	35.00
Police Quest	35.00
Aliants	24.95
Alien Fire	35.00
Santa Paravia	21.00
Lurking Horror	27.95
Star Fleet 1	39.95
Empire	39.95
Liesure Suit Larry	27.95
Gridiron	35.00
Dungeon Master	27.95
Flight Simulator	35.00
Trailblazer	27.95
••••SPECIALS••••	
Jewel of Darknss	19.95
Silicon Dreams	19.95

HARDWARE

20 meg hard disk	558.95
30 meg	749.95
60 meg	1249.95
Atari CD-ROM	499.95

National (800) 433-6122 California (800) 874-4789 (415)849-1717 Prices subject to change without notice. We ship ANYWHERE! $4.00 min S&H. No 1040's or Megas mail order. Hand delivery only, List plus $100.

CIRCLE #112 ON READER SERVICE CARD.

DATABASE DELPHI

by Andy Eddy

As you may have discovered, there's quite a range of SIG topics on Delphi: some detail various brands of computers (as we do in the Analog/Atari SIG); others reflect specific interests (such as the Micro Artists, or MANIAC SIG, as it's called). All of them are formatted similarly using the software Delphi has implemented for the system operation, but each area has been molded into a unique venue. The SIG manager(s) decides what topics will best reflect the interests of their members. In the Atari SIG, we have ours divided into specific ST and 8-bit categories that make it easier to scan through the megabytes of files and messages available.

This structure is defined by the SIG Manager and Delphi's people, so that anyone coming in will be greeted by consistency and comfort—an important consideration if you want to keep people coming back time after time. There's no doubt that many information providers' menu structures can be very cryptic, leaving the user's wallet at the mercy of the documentation, but I feel that Delphi is on the top end of all the systems when it comes to clarity, because of their employment of *English* in the menus.

Regardless of what service you use, though, the structure is generally firmly in place, and there's not much the user can contribute directly (other than suggestions) with regards to how the system is laid out. Delphi, as an exception, has one area that is entirely open to the creativity and whim of the users, allowing surveys to be created and voted on in whatever topic they choose—computer-oriented or not. Let's look at the P section in a little more depth.

At the time of this writing, the Poll area had been cleared of all but the

most recent entries. In the past, it has brought queries ranging from whether or not there is interest in buying or upgrading to a Mega ST, to what kind of games are most popular and even the feelings about the quality of the new Star Trek series. Diversity, to say the least.

If you have some time, you can garner a good cross section of data on any subject. I've also used the polls to informally research demographics in specific areas of the Atari community and used the results in articles. For example, one of the current polls is attempting to determine who is the favored candidate in the presidential campaign.

To enter the polling area, type POLL from the Analog≤ prompt. It can actually be abbreviated PO or P, as long as you make the selection unique enough that Delphi knows what you are specifying. We'll discuss this more in detail later.

Once you get the POLL≤ prompt, the menu reads:

```
BROWSE through poll results
CREATE a new poll
EDIT your poll comment
HELP
LIST poll names
RESULTS with comments
EXIT
POLL> (BROWSE, CREATE, EDIT, LIST, RESULTS, VOTE)
```

Getting to the Poll Position

If you opt to create a new poll, the prompts that Delphi gives are usually enough to get you through. The first thing asked for is the poll name. After naming the poll, you are given the choice of what kind of poll you would like: Yes/No, Multiple Choice or a range between Strongly Agree and Strongly Disagree. This provides any configuration that you may need for getting the best data possible.

After the construction process is finished, Delphi makes the survey public, adding it to the existing list for voting and optional comment. In a poll I just created (this is being written in early March), I'm trying to get an idea of which ST game titles are the most popular. For that reason, I've made the poll multiple choice so the respondents can add titles that I didn't originally place on the list. Here's what it looks like after typing RE GA (short for RESULTS GAME) from the POLL> prompt:

```
POLL> (BROWSE, CREATE, EDIT, LIST, RESULTS, VOTE) re ga
GAME HALL OF FAME, created by ANALOG2.
Creation date: FEB. 29, 1988
```

There has been a good amount of talk recently, given the release of Dungeon Master (by FTL), about some of the great gameware there is for the ST. Take some time to tell us what you think is the best in your view. You can enter your own choices, even come back later and change your mind if you see a better selection. Please note in your comment what you chose so others know what you are commenting on.

CHOICE	VOTES	PERCENT
Dungeon Master	1	25%
StarGlider	1	25%
Pawn	0	0%
Oids	1	25%
Test Drive	0	0%
Time Bandit	1	25%
Universal Military S	0	0%
TOTAL VOTE:	4	

Comments:

Time Bandit has the right stuff to make me keep coming back and back after affairs with StarGlider, Ogre, or even Gunship. I added UMS, but my vote goes to the hands-down winner—DM! While I think Dungeon Master is a great adventure, I'm not an avid adventurer. Rather, I opt for the arcade talents of Oids and its editor section that lets you add more galaxies to the ones that came with the game. A E

VOTE on this poll? (Y/N)

There's no easier way to find out user interest and have the results compiled than to use the poll software Delphi has provided. Stop by every now and then to vote on new polls, update your previous responses or add you own polls.

DELPHI NEWS

As this is being written, Delphi is finishing up the beta testing of some new features in their system software. One of those is the addition of Ymodem Batch for file transfer in your workspace and the databases. Those who send and receive files already should be familiar with Xmodem, a transfer protocol that breaks the file down into 128 byte chunks or "packets," and sends them in a hands-off operation. Each block also contains a checksum so the transmitting terminal can confirm accurate reception by the receiver. If an error is encountered, the block is automatically reset.

Similar to Xmodem, Ymodem sends the file in 1K blocks, which generally results in quicker transfers overall. An enhanced version of Ymodem—called Ymodem Batch— allows *groups* of files to be sent in 1K packets, as opposed to the manual one-by-one situation that currently exists.

Let's say you wish to download four ARCed (Archived) files that reside in your workspace. Ymodem Batch allows you to select files one at a time or with wild cards, and then start the chained transfer of *all* the files. In the above situation, YBatch (as we'll call it) is initiated, you could type YBD * ARC (which tells Delphi to YBatch Download all files with an .ARC extender) from the WS> (workspace) prompt. Of course you'll need to use terminal software that supports YBatch—the latest versions of **Flash, Interlink** and **ST-Talk Professional 2.0,** all offer YBatch compatibility.

When you tell your software to begin the transfer, the other comfort of YBatch becomes apparent: the filenames are sent automatically with the file. Folks like myself who are extremely lazy now can vegetate further in front of their terminal; better yet, some terminal software is being designed with "background transfer" ability, which functions like a printer spooler. It allows you to continue with other processes while it transfers the file from a buffer area in memory; this, in essence, enables you to double your productivity. Who said the ST can't multitask?

That about covers it for now. Till next month, C U online. . . .

APPLICATION

BASIC Draw

by Colin Faller

High Resolution Only

Many people complain about ST BASIC, saying that it's impossible to use it to create a full-fledged GEM-style program. **BASIC Draw** is an example of just what can be accomplished with ST BASIC if you're willing to apply yourself to the task. Programmers will find the menu system used in **BASIC Draw** to be of special interest.

Note that owners of either the 1040 ST or a one-megabyte 520 ST will have to alter the program slightly. This is due to a difference in the location of screen memory. All you have to do is change the statement MEM1=494560 in Line 8 to MEM1=1018848. The progam should then work with your machine.

Unfortunately, this program runs only in high resolution; so if you've been thinking about getting a monochrome monitor, now would be a great time to do it.

The program

When you run **BASIC Draw**, it'll take about three seconds to initialize, after which the main screen will be drawn and the mouse pointer will appear (in the form of a pointing finger). The screen is divided into three areas. The large area to the left is the drawing screen. The area on the right is for the menu and selected functions. The third area, located at the top of the screen, contains four boxes and a Busy box.

The first box is used for the Circle, Box, Ellipse and Polygon functions. When **BASIC Draw** is the first run the box will be white. But after selecting one of the fill styles, the pattern selected will automatically show in this box. The second box is used only in Airbrush mode. It shades in with a combination of white and black. The third and fourth boxes are used for the Erase, Line 1, Airbrush, Mirror, Circle, Box, Ellipse, Line 2 and Polygon functions. The Busy box lights when you can't use the mouse pointer.

The following is a quick overview of the functions available in **BASIC Draw**. To activate the menu, simply move the mouse pointer over it.

REVERSE	Reverses the screen from black to white or white to black
FILL 1	Fills in an area with one of the 36 GEM styles
FILL 2	Fills in an area with one of 15 Customized styles
ERASE	Used to erase small sections of your drawing
LINE 1	Lets you draw freehand in any of three-line thicknesses
AIRBRUSH	Used to shade areas of your picture
MIRROR	Turns on the mirroring option
TEXT	Allows you to put text on the drawing screen
CIRCLE	Draws a hollow or solid circle of any size
BOX	Draws a hollow or solid box of any size
ELLIPSE	Draws a hollow or solid ellipse of any size
LINE 2	Draws a straight line between two points
POLYGON	Draws a filled, multiple-sided shape
CLEAR	Clears the screen
EXIT	Returns to BASIC
LOAD/SAVE	Loads or saves a picture to disk

Now that you've got a nodding ac-

BASIC Draw

quaintance with the program, let's look at some of the features in more detail. To select a function from the menu, move the mouse pointer over your selection, and press the left mouse button (LMB).

Fills

When you select FILL 1 function you're given a choice of 36 different fill styles. To select a style, move the mouse pointer over the one you want and press the LMB. The style you have selected will be displayed in a box above the different styles and also in the top corner of the screen. To fill a part of your drawing, move the mouse pointer to the part to be filled and press the LMB. The Busy box will go on while the fill is working.

FILL 2 is the same as FILL 1, except you can choose from 15 custom fill styles.

Erase

When you choose this function, you'll first need to choose one of the eight "size boxes" at the bottom of the screen. To select an erase size, move the mouse pointer to the required size and press the LMB. Then select the color (black or white) the same way. To erase part of your drawing, move the mouse pointer to the area you want to erase and move the mouse over the area, holding down the LMB. This function can also be used as a thick drawing line.

Line 1

There are three different line thicknesses to select from. To select a line thickness move the mouse pointer to one of the three choices and press the LMB. Also select one of the two colors—black or white—in the same way. To draw with the mouse, move the mouse pointer onto the drawing screen, and when you want to draw, press the LMB. To stop drawing, release the button.

Airbrush

This function is used for shading in parts of the screen. Select one of the three different shades and press the LMB. To "spray" an area, move the mouse pointer over the area while holding down the LMB.

Mirror

You are given three different types of mirror styles. Style 1 gives you horizontal mirroring, Style 2 gives you vertical mirroring, and Style 3 gives you both. To select a mirroring style, place the mouse pointer on one of the mirror icons and press the LMB. Colors are selected in the same way. To draw, place the mouse pointer on the drawing screen and hold down the LMB. Whatever you draw will be "mirrored" in whatever manner you selected.

Text

Text is the most complicated of all of the functions. Instead of using just one menu area, it requires two: one for selecting characters and the other for selecting the size and type of characters. To switch between the two menus, press the RMB.

To select the character size and type, first press the RMB. You will then be able to choose between five character sizes and five character types. At the top of the function screen, there will be a square containing the letters "ABC." The letters show you the currently selected text style. In the middle of the menu area, there are five boxes numbered *1* to *5*. These are used to select your character size, where *1* is the smallest and *5* is the largest. To select

a character size or type move the mouse pointer over the one you've selected and press the LMB. The styles available include bold, grayed, skewed, underlined and outlined. To turn off an option, just "re-select" it.

At the bottom of the menu, you can select either black characters on a white background or white characters on a black background. To select the color, move the mouse pointer over the one you want and press the LMB.

To print the characters to the screen, return to the original menu by pressing the RMB; then move the mouse pointer onto the drawing screen. The mouse pointer will be replaced with a square cursor the size of which will depend upon the character size you selected. Place the cursor where you want the text to begin and press the LMB. The computer will calculate the number of characters that will fit on the screen from your selected position. The number will vary, of course, according to the size and style of the characters, and the location of the cursor. The maximum number of characters will be displayed at the top of the screen. To cancel the text location and choose another, move the mouse pointer over the letters ABC and press the LMB.

Now you can select the characters to print with the mouse. To display a different "page" of characters, click on one of the boxes numbered *1* to *4* with the LMB. To select a character, move the mouse pointer to the character you want and press the LMB. The character is then displayed at the top of the screen. When you have finished "typing" the characters, print them to the screen by moving the mouse pointer to the text line at the top of the screen (where the characters appeared as you selected them) and press the LMB.

BASIC Draw

Circle

You can choose between a solid or hollow circle. Move the mouse pointer to the style you want and press the LMB. If you select a solid circle, you can use any one of the three fill styles, the colors black or white or your own style. Select a fill style by moving the mouse pointer to the style you want and press the LMB.

There are three boxes at the bottom of the screen which are labeled "Plot," "Radius" and "Mouse Pos." The latter is the position of the mouse pointer. Plot is used to plot the center point of the circle. Move the mouse pointer to the center point of the circle and press the LMB. The Radius will set itself to zero. To draw the circle, move the mouse pointer to the right of the center point if you've selected a solid circle, or left or right if you've selected a hollow circle, and then press the RMB.

Box

This function works similarly to the circle function. The only difference is a change in the three boxes at the bottom of the function menu. Instead of Radius, you have a Size, and the Plot function now plots the upper left corner of the box.

Ellipse

This function is also similar to Circle, but now the radius has two values.To draw the ellipse, you can move the mouse pointer in any direction you need to get the proper size.

LINE 2

This function is used to draw a straight line between two points. This requires two operations: Plot (P) and Draw (D). First plot the starting point of your line by pressing the LMB. Select the line's endpoint with the RMB, and a line will be drawn between the two points.

Notice that, on the function menu, there are two boxes labeled *P* and *D*. When you first enter this menu, the *P* function will be activated (it'll be underlined). When you draw lines in this mode, the point you first plotted is always used as the beginning point of your line. Each time you press the RMB, a line will be drawn from the original plot point to the position of the mouse cursor. This allows you to draw multiple lines, all with the same starting point (sometimes this type of line is called a *ray*). When *D* is selected, repeated presses of the RMB will cause a line to be drawn from the previous line's ending point to the position of the mouse cursor. Using this function, you can draw complicated shapes.

Polygon

This is similar to Line 2 with the *D* option set, except it allows you to draw a *filled* shape made up of up to 63 lines. The fill style is selected in the same manner discussed previously. To fill a shape, move the mouse pointer to the box below the word "Polygon" and press the LMB.

Load and Save

To load a picture, move the mouse pointer over the Load menu selection and press the LMB. You must then choose one of the 12 files to load. Move the mouse pointer over the file you want and again press the LMB. To cancel, press the RMB.

The save function works the same way.

Colin Faller *is 21 years old and lives in the Northeast of England about a half mile from the coast. He has been programming with Atari computers for 3½ years.*

APPLICATION

High Resolution Only

Listing 1: ST BASIC

```
1 REM ************************
2 REM       BASIC  DRAW
3 REM     BY COLIN  FALLER
4 REM
5 REM      Copyright 1988
6 REM        by ST-Log
7 REM ************************
8 MEM1=494560:REM ****  One meg ST cha
nge from 494560 to 1018848  ****
9 WAVE 7:COX=570:COV=115:SZ1=13:SX%=10
:SY%=16:QMX=2:QMY=0:CX%=0:CY%=416
10 FULLW 2:GEM=SYSTAB+24:POKE GEM,1:A#
=GB:Z9=PEEK(A#+8):GOTOXY 31,15:?"  ";
11 I=INTIN:C=CONTRL:P=PTSIN:POKE C,11:
POKE C+2,2:POKE C+6,0:POKE C+10,1
12 BPX=3:RESTORE 15:FOR UV=1 TO 36:REA
D AA,BB,CC,DD,EE:COLOR 1,1,1,EE,EE
13 IF EE=4 THEN COLOR 1,1,1,4,2
14 POKE P,AA:POKE P+2,BB:POKE P+4,CC:P
OKE P+6,DD:VDISYS(1):NEXT UV
15 DATA 0,0,640,400,1,3,3,528,31,4,534
,50,634,394,4,3,387,528,394,4,0,37,530
16 DATA 383,0,533,69,637,89,1,533,2,63
7,49,0,572,145,596,157,1,533,71,637,92
17 DATA 0,533,11,617,42,1,553,11,615,4
0,4,557,15,609,34,1,559,17,611,36,0
18 DATA 533,103,637,123,1,533,105,637,
126,0,574,147,598,159,0,552,175,616
19 DATA 534,1,554,177,618,213,0,533,29
4,637,330,0,533,292,637,293,1,448,6
20 DATA 519,25,1,450,8,521,27,0,452,10
,519,25,0,360,6,411,25,1,362,8,413,27
21 DATA 0,364,10,411,25,0,260,6,311,25
,1,262,8,313,27,0,264,10,311,25,1,160
22 DATA 6,211,25,1,162,8,213,27,0,187,
10,211,25,1,164,10,187,25,0,60,6,111
23 DATA 25,1,62,8,113,27,0,64,10,111,2
5,0,77,69,78,85,569,32,66,85,83,89,456
25 POKE C,8:POKE C+2,1:POKE C+6,4:FOR
XX=1 TO 2:READ AA,BB,CC,DD,EE,FF
26 POKE I,AA:POKE I+2,BB:POKE I+4,CC:P
OKE I+6,DD:POKE P,EE:POKE P+2,FF
27 VDISYS(1):NEXT XX:POKE C,11:POKE C+
2,2:POKE C+6,0:COLOR 0,0,0
30 CC9=1:CA9=1:LX=33:LY=24
40 FOR XX=1 TO 3:READ AA,BB,CC,DD:POKE
 P,AA:POKE P+2,BB:POKE P+4,CC
42 POKE P+6,DD:VDISYS(1):NEXT XX:COLOR
 1,1,1:POKE P,455:POKE P+2,10
44 DATA 23,637,0,639,400,0,0,639,2,0,3
97,639,399,531,0,533,400,534,47,639,49
45 DATA 0,34,533,36,0,0,2,36,0,387,2,3
99,0,384,533,386
46 POKE P+4,500:POKE P+6,10:VDISYS(1):
POKE P,455:POKE P+2,25:POKE P+4,500
50 POKE P+6,25:VDISYS(1):G=PEEK(A#+12)
:?"RESOLUTION":POKE C,106:POKE C+2,0
53 POKE C+6,1:POKE I,16:VDISYS(1):POKE
 C,12:POKE C+2,1:POKE C+6,0
55 POKE P+2,12:VDISYS(1):FOR YU=32 TO
36:READ ZX$:GOTOXY YU,2:?ZX$:NEXT YU
56 FOR YU=35 TO 32 STEP-1:READ ZX$:GOT
OXY YU,4:?" ";ZX$:NEXT YU
57 DATA 8,A,S,I,C,W,A,R,D,1,2,602
58 POKE C,106:POKE C+2,0:POKE C+6,1:PO
KE I,4:VDISYS(1):POKE C,12:POKE C+2,1
60 POKE C+6,0:POKE P+2,6:VDISYS(1):GOT
OXY 33,8:?"COLIN":GOTOXY 33,9
61 ?"FALLER":POKE C,106:POKE C+2,0:POK
E C+6,1:POKE I,0:VDISYS(1):POKE C,12
62 POKE C+2,1:POKE C+6,0:POKE P+2,6:VD
ISYS(1):GOTOXY 34,6:?"BY":GOTOXY 32,16
68 FOR TU=1 TO 7:READ AA,BB,CC,DD:COLO
R 1,1,1,AA,BB:FILL CC,DD:NEXT TU
69 DATA 40,2,2,580,40,3,2,564,47,4,2,5
55,47,5,2,560,84,4,2,580,84,2,2,590,84
71 ?"530 * 344":POKE Z9,3:GEMSYS(78):P
OKE Z9,257:GEMSYS(78)
72 POKE P+2,13:VDISYS(1):POKE C,11:POK
E C+2,2:POKE C+6,0:POKE C+10,1
75 GEMSYS(79):IF PEEK(G+2)>552 AND PEE
K(G+4)<38 THEN 80 ELSE 75
80 POKE Z9,256:GEMSYS(78):POKE Z9,3:GE
MSYS(78)
81 POKE C,11:POKE C+2,2:POKE C+6,0:POK
E C+10,1:COLOR 1,1,1,4,2
82 FOR GI=0 TO 50 STEP 10:POKE P,584-G
I:POKE P+2,100-GI:POKE P+4,586+GI
83 POKE P+6,346+GI:VDISYS(1):NEXT GI:R
ESTORE 86:FOR YU=1 TO 6:READ AA,BB,CC
84 READ DD,EE,FF:COLOR 1,1,1,EE,FF:POK
E P,AA:POKE P+2,BB:POKE P+4,CC
85 POKE P+6,DD:VDISYS(1):NEXT YU:FOR U
T=74 TO 360 STEP 34:I=INTIN
86 DATA 534,50,635,395,0,534,50,634,
394,4,2,547,55,618,72,1,1,537,72,627
87 DATA 378,1,1,549,57,620,74,0,0,539,
74,629,380,0,0
90 POKE P+2,UT:POKE P+6,UT+17:VDISYS(1
):NEXT UT:GOTOXY 33,1
96 ?" MENU":POKE C,106:POKE C+2,0:POKE
 C+6,1:POKE I,4:VDISYS(1)
100 RESTORE 101:FOR UT=2 TO 18:READ XZ
$:GOTOXY 32,UT:?XZ$:NEXT UT
101 DATA REVERSE,FILL.1,FILL.2,ERASE,L
INE.1,AIRBRUSH,MIRROR TEXT,
113 DATA CIRCLE,BOX,ELLIPSE,LINE.2,POL
YGON, ,CLEAR,EXIT
114 GOTOXY 32,19:?"LOAD/SAVE":POKE I,
0:VDISYS(1)
120 POKE C,32:POKE C+2,0:POKE C+6,1:PO
KE I,3:VDISYS(1)
122 POKE C,11:POKE C+2,2:POKE C+6,0:PO
KE C+10,1:COLOR 1,1,1,1,1
191 POKE Z9,257:GEMSYS(78)
192 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
194 IF W>540 AND W<629 AND X>74 AND X<
380 THEN 199 ELSE 192
199 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
200 IF X>363 THEN X1=363:GOTO 299
201 IF X>346 THEN X1=346:GOTO 299
202 IF X>329 THEN X1=329:GOTO 299
203 IF X>312 THEN 199
204 IF X>295 THEN X1=295:GOTO 299
205 IF X>278 THEN X1=278:GOTO 299
206 IF X>261 THEN X1=261:GOTO 299
207 IF X>244 THEN X1=244:GOTO 299
208 IF X>227 THEN X1=227:GOTO 299
210 IF X>210 THEN 199
211 IF X>193 THEN X1=193:GOTO 299
212 IF X>176 THEN X1=176:GOTO 299
213 IF X>159 THEN X1=159:GOTO 299
214 IF X>142 THEN X1=142:GOTO 299
215 IF X>125 THEN X1=125:GOTO 299
```

```
216 IF X>108 THEN X1=108:GOTO 299
217 IF X>91 THEN X1=91:GOTO 299
218 X1=74!GOTO 299
250 POKE C,11:POKE C+2,2:POKE C+6,0:PO
KE C+10,1
255 POKE P,0:POKE P+2,37:POKE P+4,530:
POKE P+6,383:VDISYS(1)
256 POKE C,32:POKE C+2,0:POKE C+6,1:PO
KE I,1:VDISYS(1):GOTO 81
299 POKE Z9,256:GEMSYS(78):POKE P,539:
POKE P+2,X1!POKE P+6,X1+17
300 POKE P+4,629:VDISYS(1):LINEF 538,X
1-38,538,X1-38:POKE Z9,257:GEMSYS(78)
301 GEMSYS(79):W=PEEK(G+2):X=PEEK(G+4)
:IF PEEK(G+6)=1 THEN 304
302 IF X>X1 AND X<X1+18 AND W>540 AND
W<629 THEN GOTO 301
303 POKE Z9,256:GEMSYS(78):VDISYS(1):L
INEF 538,X1-38,538,X1-38:GOTO 191
304 C=CONTRL:I=INTIN:P=PTSIN:IF X1=363
 THEN GOTO SAV
305 IF X1=74 THEN POKE Z9,256:GEMSYS(7
8):GOTO 250
306 IF X1=91 THEN GOSUB 330:?"  FILL.1
":GOTO FILLX
307 IF X1=108 THEN GOSUB 330:?"  FILL.
2":GOTO FIL
308 IF X1=125 THEN GOSUB 330:?"  ERASE
":GOTO ERASEX
309 IF X1=142 THEN GOSUB 330:?"  LINE.
1":GOTO LINEX
310 IF X1=159 THEN GOSUB 330:?" AIR8RU
SH":GOTO AIRX
311 IF X1=176 THEN GOSUB 330:?"  MIRRO
R":GOTO MIRRORX
312 IF X1=193 THEN GOSUB 330:?"   TEXT
":GOTO TEXTX
313 IF X1=227 THEN GOSUB 330:?"  CIRCL
E":GOTO CIRCLEX
314 IF X1=244 THEN GOSUB 330:?"   BOX"
:GOTO 80XX
315 IF X1=261 THEN GOSUB 330:?" ELLIPS
E":GOTO ELLEX
316 IF X1=278 THEN GOSUB 330:?"  LINE.
2":GOTO LINZ
317 IF X1=295 THEN GOSUB 330:?" POLYGO
N":GOTO POLY
318 IF X1=329 THEN ZIP1=1:GOTO ALERT
319 IF X1=346 THEN ZIP1=2:GOTO ALERT
320 COLOR 0,0,0,0,0:POKE P,500:POKE P+
2,15:POKE P+6,20
321 POKE P+6,20:VDISYS(1):COLOR 1,1,1,
1,1:RETURN
323 CLEARX:POKE C,32:POKE C+2,0:POKE C
+6,1:POKE I,1:VDISYS(1)
324 COLOR 1,1,1,0,0:POKE C,11:POKE C+2
,2:POKE C+6,0
325 FOR 8R=1 TO 172 STEP 10:POKE P,171
-8R:POKE P+2,208-8R:POKE P+4,359+8R
326 POKE P+6,212+8R:VDISYS(1):NEXT 8R:
GOTO 81
330 POKE C,32:POKE C+2,0:POKE C+6,1:PO
KE I,1:VDISYS(1):POKE Z9,256
331 GEMSYS(78):POKE I,0:VDISYS(1):POKE
 C,11:POKE C+2,2:POKE C+6,0
332 POKE C+10,1:POKE P,500:POKE P+2,15
:POKE P+4,515:POKE P+6,20:VDISYS(1)
333 COLOR 1,1,1,4,2:FOR GI=0 TO 50 STE
P 10:POKE P,584-GI:POKE P+2,100-GI
334 POKE P+4,586+GI:POKE P+6,346+GI:VD
ISYS(1):NEXT GI:GOTOXY 32,1
335 RESTORE 336:FOR YU=1 TO 4:READ AA,
88,CC,DD,EE,FF:COLOR 1,1,1,EE,FF
336 DATA 534,50,635,395,4,2,534,50,634
,394,4,2,539,55,626,72,1,1
337 DATA 541,57,628,74,0,0
338 POKE P,AA:POKE P+2,BB:POKE P+4,CC:
POKE P+6,DD:VDISYS(1):NEXT YU:RETURN
339 IF W>264 AND W<311 AND X>10 AND X<
25 AND OG=1 THEN CC9=1:CA9=1:GOTO 343
341 IF W>364 AND W<411 AND X>10 AND X<
25 AND OG=1 THEN CC9=0:CA9=0:GOTO 343
342 RETURN
343 POKE C,11:POKE C+2,2:POKE C+6,0:CO
LOR 1,1,1,CC9,CA9:POKE P,562-ZX1
344 POKE P+2,90+ZX2:POKE P+4,609-ZX1:P
OKE P+6,105+ZX2:VDISYS(1):RETURN
351 FILLX:RESTORE 358:FOR UY=0 TO 9:RE
AD AA,88,CC,DD,EE,FF:COLOR 1,1,1,EE,FF
355 POKE P,AA:POKE P+2,88:POKE P+4,CC:
POKE P+6,DD:VDISYS(1):NEXT UY
358 DATA 538,80,629,80,1,1,540,82,631
,391,0,0,545,87,626,142,1,1,545,87
359 DATA 624,140,4,2,541,147,630,147,0
,0,549,91,618,134,1,1,551,93,620,136
360 DATA 0,0,543,150,570,168,0,0,572,1
50,599,168,1,2,601,150,628,168,2,2
362 U=3:B=2:FOR Z=170 TO 380 STEP 20:F
OR V=543 TO 625 STEP 29
364 COLOR 1,1,1,U,B:U=U+1:POKE P,V:POK
E P+2,Z:POKE P+4,V+27
365 POKE P+6,Z+18:VDISYS(1):IF U=25 TH
EN U=1:B=3
366 NEXT Z:NEXT V:I=4:GOSUB 320
367 COLOR 1,1,1,FF1,GFG:COLOR 1,1,1,FF
1,GFG:POKE P,553
368 POKE P+2,95:POKE P+4,618:POKE P+6,
134:VDISYS(1)
369 POKE Z9,257:GEMSYS(78)
370 GEMSYS(79):W=PEEK(G+2):X=PEEK(G+4)
:OG=PEEK(G+6)
372 IF W>552 AND X<38 THEN 80
373 IF W<530 AND X>37 AND X<383 AND OG
=I THEN 420
374 IF OG=I AND W>541 AND W<627 AND X>
147 THEN 400
375 I=1:IF W<530 AND X>37 AND X<383 TH
EN POKE Z9,5:GEMSYS(78):GOTO 370
376 POKE Z9,3:GEMSYS(78):GOTO 370
400 IF W<575 AND X<167 THEN L8=0:GOTO
432
401 IF W>575 AND X<167 THEN L8=0:GOTO
431
402 IF X<187 THEN L8=3:GOTO 431
403 IF X<207 THEN L8=6:GOTO 431
404 IF X<227 THEN L8=9:GOTO 431
405 IF X<247 THEN L8=12:GOTO 431
406 IF X<267 THEN L8=15:GOTO 431
407 IF X<287 THEN L8=18:GOTO 431
408 IF X<307 THEN L8=21:GOTO 431
409 IF W>575 AND X<327 THEN L8=0:GOTO
430
410 IF X<327 THEN L8=24:GOTO 431
411 IF X<347 THEN L8=3:GOTO 430
412 IF X<367 THEN L8=6:GOTO 430
413 IF X<387 THEN L8=9:GOTO 430 ELSE 3
70
420 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:ZV=529:POKE Z9,256:GEMSYS(78)
421 COLOR 1,1,1,1,1:IF S1=W AND S2=X T
HEN 369
422 POKE P,500:POKE P+2,15:POKE P+4,51
5:POKE P+6,20:VDISYS(1):S1=W:S2=X:I=3
424 COLOR 1,1,1,FF1,GFG:COLOR 1,1,1,FF
1,GFG:FILL W-1,X-38:LINEF ZV,0,ZV,344
426 COLOR 0,0,0,0,0:VDISYS(1):GOTO 369
430 GFG=3:GOTO 440
431 GFG=2:GOTO 440
432 GFG=0:GOTO 440
440 COLOR 1,1,1,FF1,GFG:GEMSYS(79):W=P
EEK(g+2)-541:X=PEEK(g+4):KF=W/29
441 FF1=LB+KF:COLOR 1,1,1,FF1,GFG:DW=0
442 POKE P,553:POKE P+2,95:POKE P+4,61
8:POKE P+6,134:VDISYS(1)
443 POKE P,64:POKE P+2,10:POKE P+4,111
:POKE P+6,25:VDISYS(1):GOTO 370
```

```
450 ERASEX:RESTORE 451:HH=534:THE=19:G
OSUB 800:ZX1=0:ZX2=0:GOSUB 320
451 DATA 24,86,75,105,1,26,88,77,107,0
,6,206,91,373,1,8,208,93,375,0,10
452 DATA 210,50,373,0,51,210,91,373,0,
10,250,90,251,0,10,291,90,292,0,10,332
453 DATA 90,333,0,28,228,32,232,0,67,2
26,75,234,0,24,264,36,276,0,63,262,79
454 DATA 278,0,20,302,40,322,0,59,300,
83,324,0,16,339,44,367,0,55,337,87,369
455 DATA 0,26,131,72,175,1,28,133,74,1
77,0
456 COLOR 1,1,1,0,0:POKE P,564:POKE P+
2,135:POKE P+4,606:POKE P+6,175
457 VDISYS(1):POKE P,585-B0:POKE P+2,1
55-B0:POKE P+4,585+B0
458 POKE P+6,155+B0:VDISYS(1):GOSUB 34
3
461 POKE Z9,257:GEMSYS(78)
462 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:OG=PEEK(G+6)
465 IF W<530 AND X>37 AND X<383 AND OG
=1 THEN GOTO 500
466 IF W<530 AND X>37 AND X<383 THEN P
OKE Z9,7:GEMSYS(78):GOTO 462
467 IF W>552 AND X<38 THEN 80
471 GOSUB 339:IF OG=1 AND W>544 AND W<
625 AND X>210 AND X<373 THEN 475
472 POKE Z9,3:GEMSYS(78):GOTO 462
475 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
476 IF W>585 AND X>333 THEN B0=16:GOTO
 485
477 IF W<585 AND X>333 THEN B0=14:GOTO
 485
478 IF W>585 AND X>292 THEN B0=12:GOTO
 485
479 IF W<584 AND X>292 THEN B0=10:GOTO
 485
480 IF W>585 AND X>251 THEN B0=8:GOTO
485
481 IF W<584 AND X>251 THEN B0=6:GOTO
485
482 IF W>585 AND X>210 THEN B0=4:GOTO
485
483 IF W<584 AND X>210 THEN B0=2:GOTO
485 ELSE 462
485 COLOR 1,1,1,0,0:POKE P,564:POKE P+
2,135:POKE P+4,606:POKE P+6,175
486 VDISYS(1):POKE P,585-B0:POKE P+2,1
55-B0:POKE P+4,585+B0:POKE P+6,155+B0
488 VDISYS(1):GOTO 462
500 COLOR CC9,CC9,CC9,CC9,CC9
501 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:IF PEEK(G+6)=0 THEN 462
511 IF W<1+B0 THEN W=1+B0
512 IF W>529-B0 THEN W=529-B0
513 IF X<38+B0 THEN X=38+B0
514 IF X>382-B0 THEN X=382-B0
520 POKE P,W-B0:POKE P+2,X-B0:POKE P+4
,W+B0:POKE P+6,X+B0
530 POKE Z9,256:GEMSYS(78):VDISYS(1):P
OKE Z9,257:GEMSYS(78):GOTO 501
600 LINEX:RESTORE 610:HH=534:THE=12:GO
SUB 800:ZX1=2:ZX2=12:GOSUB 320
610 DATA 22,98,73,117,1,24,100,75,119,
0,7,150,89,178,1,9,152,91,180,0,7,256
611 DATA 89,348,1,9,258,91,350,0,11,26
0,89,289,0,11,290,89,319,0,11,320,89
613 DATA 348,0,20,274,80,274,0,20,304,
80,305,0,20,333,80,335,1
614 COLOR 1,1,1,0,0:POKE P,545:POKE P+
2,154:POKE P+4,623:POKE P+6,178
615 VDISYS(1):COLOR 1,1,1,1,1:POKE P,5
54:POKE P+2,166-LO:POKE P+4,614
616 POKE P+6,166+LO:VDISYS(1):GOSUB 34
3
620 POKE Z9,257:GEMSYS(78)
621 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:OG=PEEK(G+6)
624 IF W>544 AND W<624 AND X>259 AND X
<349 AND OG=1 THEN 630
625 IF W<530 AND X>37 AND X<383 AND OG
=1 THEN GOTO 650
626 IF W<530 AND X>37 AND X<383 THEN P
OKE Z9,5:GEMSYS(78):GOTO 621
627 IF W>552 AND X<38 THEN 80
629 GOSUB 339:POKE Z9,3:GEMSYS(78):GOT
O 621
630 GEMSYS(79):X=PEEK(g+4):IF X<349 TH
EN LO=1
638 IF X<319 THEN LO=0.5
639 IF X<289 THEN LO=0
640 COLOR 1,1,1,0,0:POKE P,545:POKE P+
2,154:POKE P+4,623:POKE P+6,178
642 VDISYS(1):COLOR 1,1,1,1,1:POKE P,5
54:POKE P+2,166-LO:POKE P+4,614
643 POKE P+6,166+LO:VDISYS(1):GOTO 621
649 POKE Z9,257:GEMSYS(78):GOTO 621
650 IF LO=0 THEN POKE Z9,256:GEMSYS(78
):GOTO 660
651 IF LO=0.5 THEN POKE Z9,256:GEMSYS(
78):GOTO 670
652 IF LO=1 THEN POKE Z9,256:GEMSYS(78
):GOTO 680 ELSE 621
660 Y=W-1:Z=X-38:COLOR CC9,CC9,CC9
661 GEMSYS(79):W=PEEK(G+2)-1:X=PEEK(G+
4)-38
662 IF PEEK(G+6)=0 THEN 649
663 IF W>528 THEN 665
664 LINEF W,X,Y,Z:Y=W:Z=X:GOTO 661
665 LINEF 528,X,Y,Z
666 GEMSYS(79):W=PEEK(G+2)-1:X=PEEK(G+
4)-38:IF PEEK(G+6)=0 THEN 649
668 IF W>528 THEN 666 ELSE Y=528:Z=X:G
OTO 661
670 Y=W-1:Z=X-38:COLOR CC9,CC9,CC9
671 GEMSYS(79):W=PEEK(G+2)-1:X=PEEK(G+
4)-38:IF PEEK(G+6)=0 THEN 649
672 IF W>528 THEN W=528
673 LINEF W,X,Y,Z:LINEF W-1,X-1,Y-1,Z-
1:LINEF W-1,X,Y-1,Z:LINEF W,X-1,Y,Z-1
674 IF W=528 THEN GOTO 675 ELSE Y=W:Z=
X:GOTO 671
675 GEMSYS(79):W=PEEK(G+2)-1:X=PEEK(G+
4)-38:IF PEEK (G+6)=0 THEN 649
676 IF W>529 THEN 675:Y=528:Z=X:GOTO 6
71
680 Y=W-2:Z=X-39:COLOR CC9,CC9,CC9
681 GEMSYS(79):W=PEEK(G+2)-2:X=PEEK(G+
4)-39:IF PEEK(G+6)=0 THEN 649
682 IF W>526 THEN W=526
683 IF W<1 THEN W=-3
684 LINEF W,X,Y,Z:LINEF W+2,X+2,Y+2,Z+
2:LINEF W+2,X,Y+2,Z
685 LINEF W,X+2,Y,Z+2:LINEF W+1,X,Y+1,
Z:LINEF W+1,X+2,Y+1,Z+2
686 LINEF W,X+1,Y,Z+1:LINEF W+2,X+1,Y+
2,Z+1:LINEF W+1,X+1,Y+1,Z+1
687 IF W=526 THEN 688 ELSE Y=W:Z=X:GOT
O 681
688 GEMSYS(79):W=PEEK(G+2)-1:X=PEEK(G+
4)-38:IF PEEK (G+6)=0 THEN 649
689 IF W>527 THEN 688:Y=526:Z=X:GOTO 6
81
701 AIRX:HH=534:RESTORE 704:THE=2:GOSU
B 800:GOSUB 320
704 DATA 22,98,73,117,1,24,100,75,119,
0
705 COLOR 1,1,1,C1,C1:POKE P,560:POKE
P+2,102:POKE P+4,607:POKE P+6,117
706 VDISYS(1):COLOR C2,C2,C2,C2,C2:POK
E P,584:POKE P+2,103:POKE P+4,606
707 POKE P+6,116:VDISYS(1)
720 POKE Z9,257:GEMSYS(78)
721 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:OG=PEEK(G+6)
```

```
722 IF W>552 AND X<38 THEN 80
725 IF W<530 AND X>37 AND X<383 AND OG
=1 THEN GOTO 735
726 IF W<530 AND X>37 AND X<383 THEN P
OKE Z9,5:GEMSYS(78):GOTO 721
727 IF W>164 AND W<211 AND X>10 AND X<
25 AND OG=1 THEN C1=0:C2=1:GOTO 731
728 IF W>264 AND W<311 AND X>10 AND X<
25 AND OG=1 THEN C1=1:C2=1:GOTO 731
729 IF W>364 AND W<411 AND X>10 AND X<
25 AND OG=1 THEN C1=0:C2=0:GOTO 731
730 POKE Z9,3:GEMSYS(78):GOTO 721
731 COLOR 1,1,1,C1,C1:POKE P,560:POKE
P+2,102:POKE P+4,607:POKE P+6,117
732 VDISYS(1):COLOR C2,C2,C2,C2,C2:POK
E P,584:POKE P+2,103:POKE P+4,606
734 POKE P+6,116:VDISYS(1):GOTO 721
735 GEMSYS(79):W=PEEK(G+2)-6:X=PEEK(G+
4)-43:Z1=RND*10
736 IF PEEK(G+6)=0 THEN 721
737 IF W>519 THEN W=519
738 Z2=RND*10:Z3=RND*10:Z4=RND*10:Z5=R
ND*10:COLOR C1,C1,C1:POKE Z9,256
739 GEMSYS(78):LINEF Z1+W,Z2+X,Z1+W,Z2
+X:LINEF Z5+W,Z1+X,Z5+W,Z1+X
740 LINEF Z2+W,Z5+X,Z2+W,Z5+X:COLOR C2
,C2,C2:LINEF Z3+W,Z4+X,Z3+W,Z4+X
741 LINEF Z4+W,Z2+X,Z4+W,Z2+X:LINEF Z5
+W,Z3+X,Z5+W,Z3+X:POKE Z9,257
742 GEMSYS(78):GOTO 735
750 MIRRORX:RESTORE 751:HH=534:THE=13:
GOSUB 800:ZX1=2:ZX2=12:GOSUB 320
751 DATA 22,98,73,117,1,24,100,75,119,
0,13,136,83,188,1,15,138,85,190,0,13
752 DATA 222,83,378,1,15,224,85,380,0,
17,277,83,327,0,17,226,83,275,0,17,329
753 DATA 83,378,0,50,226,50,275,0,17,3
02,83,352,0,17,354,83,354,0,50,329,50
754 DATA 378,0
755 COLOR 1,1,1,0,0:POKE P,551:POKE P+
2,140:POKE P+4,617:POKE P+6,188
756 VDISYS(1):LINEF 550+LX,102,550+LX,
150:LINEF 550,102+LY,616,102+LY
757 GOSUB 343
758 POKE Z9,257:GEMSYS(78)
759 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:OG=PEEK(G+6)
762 IF W>551 AND W<619 AND X>224 AND X
<380 AND OG=1 THEN 775
763 IF W<530 AND X>37 AND X<383 AND OG
=1 THEN GOTO 784
764 IF W<530 AND X>37 AND X<383 THEN P
OKE Z9,5:GEMSYS(78):GOTO 759
765 IF W>552 AND X<38 THEN 80
767 GOSUB 339:POKE Z9,3:GEMSYS(78):GOT
O 759
775 GEMSYS(79):X=PEEK(G+4)
776 IF X<276 THEN 780
777 IF X<328 THEN 779
778 LX=33:LY=24:W1=0:W2=0:X3=0:X2=0:GO
TO 781
779 LX=0:LY=24:W1=0:W2=640:X3=400:X2=0
:GOTO 781
780 LX=33:LY=0:W1=640:W2=0:X3=0:X2=400
781 COLOR 1,1,1,0,0:POKE P,551:POKE P+
2,140:POKE P+4,617:POKE P+6,188
782 VDISYS(1):LINEF 550+LX,102,550+LX,
150:LINEF 550,102+LY,616,102+LY
783 GOTO 759
784 POKE Z9,256:GEMSYS(78):Y=W-1:Z=X-3
8:COLOR CC9,CC9,CC9
785 GEMSYS(79):W=PEEK(G+2)-1:X=PEEK(G+
4)-38:IF PEEK(G+6)=0 THEN 758
786 IF W>528 THEN W=528
787 LINEF 527-W+W2,X,527-Y+W2,Z:LINEF
527-W+W1,344-X+X3,527-Y+W1,344-Z+X3
788 LINEF W,X,Y,Z:LINEF W-W1,344-X+X2,
Y-W1,344-Z-X2:Y=W:Z=X
789 IF W=528 THEN 792
790 IF W<1 THEN 794 ELSE 785
792 GEMSYS(79):W=PEEK(G+2)-1:Z=PEEK(G+
4)-38:IF W<528 THEN 785
793 IF PEEK(G+6)=0 THEN 758 ELSE 792
794 GEMSYS(79):W=PEEK(G+2)-1:Z=PEEK(G+
4)-38:IF W>1 THEN 785
795 IF PEEK(G+6)=0 THEN 758 ELSE 794
798 RESTORE 802:HH=534:GOTO 800
799 RESTORE 805:HH=534
800 FOR GOZ=1 TO THE:READ AA,BB,CC,DD,
EE:COLOR 1,1,1,EE,EE:POKE P,AA+HH
801 POKE P+2,BB:POKE P+4,CC+HH:POKE P+
6,DD:VDISYS(1):NEXT GOZ:RETURN
802 DATA 5,326,85,380,1,7,328,87,382,0
,7,346,87,363,0,5,275,74,312,1,7,277
803 DATA 76,314,0,7,295,77,295,0,5,207
,66,260,1,7,209,68,262,0,7,227,68,244
804 DATA 0
805 DATA 5,145,46,176,1,7,147,48,178,0
,53,145,92,176,1,55,147,94,178,0,25,89
806 DATA 74,130,1,27,91,76,132,0,5,326
,85,380,1,7,328,87,382,0,7,346,87,363
807 DATA 0,5,258,75,312,1,7,260,77,314
,0,7,278,77,295,0,5,190,67,244,1,7,192
808 DATA 69,246,0,7,210,69,227,0,12,15
2,43,173,0,60,152,89,173,1,32,96,71
809 DATA 127,0,36,100,67,123,0
813 CIRCLEX:Y=0:Z=38:BG=0:TH=0:THE=15:
GOSUB 798:GOSUB 320:COLOR 1,1,1,0,0
814 POKE C,11:POKE C+2,3:POKE C+6,0:PO
KE C+10,4:POKE P,561:POKE P+2,162
815 POKE P+8,10:VDISYS(1):COLOR 1,1,1,
1,1:POKE P,608:VDISYS(1):RESTORE 817
816 FOR IU=1 TO 8:READ AA,B$:GOTOXY 32
,AA:?B$;:NEXT IU:GOTO 821
817 DATA 10,PLOT:11,X 0,12,Y 0,14,RADI
US,15,Z 0,16,MOUSE POS,18,X 0,19,Y 0
820 POKE Z9,256:GEMSYS(78)
821 COLOR 1,1,1,TH,TH:POKE P,585:POKE
P+2,111:POKE P+8,16:VDISYS(1)
822 POKE P+8,10:VDISYS(1):C1=1:C2=1
825 POKE Z9,257:GEMSYS(78)
828 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:OG=PEEK(G+6)
829 IF W<530 AND X>36 AND X<384 AND OG
=2 THEN 845
830 IF W<530 AND X>36 AND X<384 AND OG
=1 THEN 839
831 IF W<530 AND X>37 AND X<383 THEN P
OKE Z9,5:GOTO 837
834 IF W>552 AND X<38 THEN 80
835 IF OG=1 THEN 875
836 POKE Z9,3:GEMSYS(78):GOTO 828
837 GEMSYS(78):GOTOXY 32,18:?"W-1" "
:GOTOXY 32,19:?"Y-38" ";
838 GOTOXY 32,15:?"Z-BG-1" ":GOTO 82
8
839 POKE Z9,256:GEMSYS(78):GOTOXY 32,1
1:?"X-1" ":GOTOXY 32,12
840 ?"Y-38" ":LINEF W-1,X-38,W-1,X-3
8:BG=W-1:Y=W:Z=X:GOTO 825
845 POKE Z9,256:GEMSYS(78):IF BG+BG-W>
527 THEN 825
846 IF TH=1 THEN 869
847 CIRCLE Y-1,Z-38,W-Y:GOTO 825
869 IF BG+BG-W<-1 THEN 825
870 IF BG-W+Z<37 THEN 825
871 IF BG-W>0 THEN 825
872 IF BG-W-Z<-383 THEN 825
874 POKE P,Y:POKE P+2,Z:POKE P+8,W-Y:U
DISYS(1):GOTO 825
875 IF W>589 AND X>151 AND X<182 AND W
<628 THEN TH=1:GOTO 820
876 IF W>543 AND X>151 AND X<182 AND W
<682 THEN TH=0:GOTO 820
877 IF TH=0 THEN 828
878 GOSUB 880:GOTO 884
```

```
880 IF W)64 AND W(111 AND X)10 AND X(2
5 THEN C1=FF1:C2=GFG
881 IF W)264 AND W(311 AND X)10 AND X(
25 THEN C1=1:C2=1
882 IF W)364 AND W(411 AND X)10 AND X(
25 THEN C1=0:C2=0
883 COLOR 1,1,1,C1,C2:COLOR 1,1,1,C1,C
2:RETURN
884 POKE P,585:POKE P+2,111:POKE P+8,1
6:VDISYS(1):GOTO 828
985 BOXX:Y=0:Z=0:BG=0:BE=38:TH=0:THE=1
9:GOSUB 799:GOSUB 320:C1=1:C2=1
906 RESTORE 907:FOR IU=1 TO 9:READ AA,
B$:GOTOXY 32,AA:?B$;:NEXT IU:GOTO 911
907 DATA 9,PLOT,10,X 0,11,Y 0,13,SIZE,
14,X 0,15,Y 0,17,MOUSE POS,18,X,19,Y
910 POKE Z9,256:GEMSYS(78)
911 COLOR 1,1,1,TH,TH:POKE P,566:POKE
P+2,96:POKE P+4,605:POKE P+6,127
912 VDISYS(1):POKE P,570:POKE P+2,100:
POKE P+4,600:POKE P+6,123:VDISYS(1)
915 POKE Z9,257:GEMSYS(78)
918 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:OG=PEEK(G+6)
919 IF W(530 AND X)36 AND X(384 AND OG
=2 THEN 931
920 IF W(530 AND X)36 AND X(384 AND OG
=1 THEN 929
921 IF W(530 AND X)37 AND X(383 THEN P
OKE Z9,5:GOTO 927
924 IF W)552 AND X(38 THEN 80
925 IF OG=1 THEN 940
926 POKE Z9,3:GEMSYS(78):GOTO 918
927 GEMSYS(78):GOTOXY 32,18:?"X"W-1" "
:GOTOXY 32,19:?"Y"X-38" ";
928 GOTOXY 32,14:?"X"W-BG-1" ":GOTOXY
32,15:?"Y"X-BE" ":GOTO 918
929 POKE Z9,256:GEMSYS(78):GOTOXY 32,1
8:?"X"W-1" ":GOTOXY 32,11
930 ?"Y"X-38" ":LINEF W-1,X-38,W-1,X-3
8:BG=W-1:BE=X:Y=W-1:Z=X-38:GOTO 915
931 POKE Z9,256:GEMSYS(78):IF TH=1 THE
N 935
933 LINEF Y,Z,W-1,Z:LINEF Y,Z,Y,X-38
934 LINEF Y,X-38,W-1,X-38:LINEF W-1,Z,
W-1,X-38:GOTO 915
935 POKE P,Y+1:POKE P+2,Z+38:POKE P+4,
W:POKE P+6,X:VDISYS(1):GOTO 915
940 IF W)589 AND X)147 AND X(178 AND W
(628 THEN TH=1:GOTO 910
942 IF W)543 AND X)147 AND X(178 AND W
(682 THEN TH=0:GOTO 910
943 IF TH=0 THEN 918
945 GOSUB 880
948 POKE P,566:POKE P+2,96:POKE P+4,60
5:POKE P+6,127:VDISYS(1):GOTO 918
950 ELLEX:Y=0:Z=0:BG=0:BE=38:TH=0:THE=
15:GOSUB 799:GOSUB 320:COLOR 1,1,1,0,0
953 RESTORE 954:FOR IU=1 TO 9:READ AA,
B$:GOTOXY 32,AA:?B$;:NEXT IU:POKE C,11
954 DATA 9,PLOT,10,X 0,11,Y 0,13,RADIU
S,14,X 0,15,Y 0,17,MOUSE POS,18,X,19,Y
955 POKE C+2,2:POKE C+6,0:POKE C+10,5:
POKE P,561:POKE P+2,162:POKE P+4,14
956 POKE P+6,10:VDISYS(1):COLOR 1,1,1,
1,1:POKE P,608:VDISYS(1):GOTO 961
960 POKE Z9,256:GEMSYS(78)
961 COLOR 1,1,1,TH,TH:POKE P,585:POKE
P+2,111:POKE P+4,18:POKE P+6,14
962 VDISYS(1):POKE P+4,14:POKE P+6,10:
VDISYS(1):C1=1:C2=1
965 POKE Z9,257:GEMSYS(78)
968 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4)
:OG=PEEK(G+6)
969 IF W(530 AND X)36 AND X(384 AND OG
=2 THEN 981
970 IF W(530 AND X)36 AND X(384 AND OG
=1 THEN 979
971 IF W(530 AND X)37 AND X(383 THEN P
OKE Z9,5:GOTO 977
974 IF W)552 AND X(38 THEN 80
975 IF OG=1 THEN 990
976 POKE Z9,3:GEMSYS(78):GOTO 968
977 GEMSYS(78):GOTOXY 32,18:?"X"W-1" "
:GOTOXY 32,19:?"Y"X-38" ";
978 GOTOXY 32,14:?"X"W-BG-1" ":GOTOXY
32,15:?"Y"X-BE" ":GOTO 968
979 POKE Z9,256:GEMSYS(78):GOTOXY 32,1
8:?"X"W-1" ":GOTOXY 32,11
980 ?"Y"X-38" ":LINEF W-1,X-38,W-1,X-3
8:BG=W-1:BE=X:Y=W-1:Z=X-38:GOTO 965
981 POKE Z9,256:GEMSYS(78):IF TH=1 THE
N 983
982 IF BG+BG-W)528 THEN 965 ELSE ELLIP
SE Y,Z,W-Y,X-Z-38:GOTO 965
983 IF BG+BG-W(-1 THEN 965
984 IF BE-X+Z(-1 THEN 965
985 IF BG-W)0 THEN 965
986 IF BE-X+Z)344 THEN 965
988 POKE P,Y+1:POKE P+2,Z+38:POKE P+4,
W-Y-1:POKE P+6,X-Z-38:VDISYS(1)
989 GOTO 965
990 IF W)589 AND X)147 AND X(178 AND W
(628 THEN TH=1:GOTO 960
992 IF W)543 AND X)147 AND X(178 AND W
(682 THEN TH=0:GOTO 960
993 IF TH=0 THEN 968 ELSE GOSUB 880
994 POKE P,585:POKE P+2,111:POKE P+4,1
8:POKE P+6,14:VDISYS(1):GOTO 968
996 DATA 538,80,629,249,1,1,540,82,631
,251,0,0,545,87,626,142,1,1,545,87,624
997 DATA 140,4,2,541,147,630,147,0,0,5
49,91,618,134,1,1,551,93,620,136,0,0
1000 LINZ:V=0:Z=0:THE=13:HH=534:RESTOR
E 1005:GOSUB 800:GOSUB 320:BG=0:BE=38
1005 DATA 5,326,85,380,1,7,328,87,382,
0,7,346,87,363,0,5,258,75,312,1,7,260
1006 DATA 77,314,0,7,278,77,295,0,5,19
0,67,244,1,7,192,69,246,0,7,210,69,227
1007 DATA 0,22,98,73,117,1,24,100,75,1
19,0,11,140,87,158,1,13,142,89,160,0
1008 GOTOXY 33,6:?"P D":LINEF 556,1
20,573,120:LINEF 585,105,585,122
1009 FOR IU=1 TO 9:READ AA,B$:GOTOXY 3
2,AA:?B$;:NEXT IU:ZX1=2:ZX2=12:LPL=2
1010 DATA 9,PLOT,10,X 0,11,Y 0,13,DRAW
,14,X 0,15,Y 0,17,MOUSE POS,18,X,19,Y
1014 L8=556:GOSUB 543
1015 POKE Z9,257:GEMSYS(78):COLOR 1,1,
1
1016 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):OG=PEEK(G+6)
1017 IF W(530 AND X)36 AND X(384 AND O
G=2 THEN 1060
1019 IF W(530 AND X)36 AND X(384 AND O
G=1 THEN 1029
1020 IF W)547 AND W(623 AND X)142 AND
X(160 AND OG=1 THEN 1050
1021 IF W(530 AND X)37 AND X(383 THEN
POKE Z9,7:GOTO 1027
1022 IF W)552 AND X(38 THEN 80
1026 GOSUB 339:POKE Z9,3:GEMSYS(78):GO
TO 1016
1027 GEMSYS(78):GOTOXY 32,18:?"X"W-1"
":GOTOXY 32,19:?"Y"X-38" "
1028 GOTOXY 32,14:?"X"W-BG-1" ":GOTOXY
32,15:?"Y"X-BE" ":GOTO 1016
1029 POKE Z9,256:GEMSYS(78)
1030 COLOR 1,1,1:GOTOXY 32,18:?"X"W-1"
":GOTOXY 32,11:?"Y"X-38" ":BG=W-1
1031 COLOR CC9,CC9,CC9:LINEF W-1,X-38,
W-1,X-38:BE=X:Y=W-1:Z=X-38:GOTO 1015
1032 POKE Z9,256:GEMSYS(78):COLOR CC9,
CC9,CC9:LINEF W-1,X-38,Y,Z:RETURN
1050 POKE Z9,256:GEMSYS(78):IF W(586 T
HEN L8=556:L9=595
```

```
1051 COLOR 0,0,0:IF W>586 THEN L8=595:
L9=556
1052 LINEF L9,120,L9+17,120:COLOR 1,1,
1:LINEF L8,120,L8+17,120:GOTO 1015
1060 IF L8=595 THEN GOSUB 1032:GOTO 10
30 ELSE GOSUB 1032:GOTO 1015
1099 FILE:RESTORE 996:FOR UV=1 TO 7:REA
D AA,BB,CC,DD,EE,FF:COLOR 1,1,1,EE,FF
1100 POKE P,AA:POKE P+2,BB:POKE P+4,CC
:POKE P+6,DD:VDISYS(1):NEXT UV
1101 RESTORE 1160:FOR Z=150 TO 233 STE
P 20:FOR Y=34 TO 225 STEP 29
1102 POKE C,112:POKE C+2,0:POKE C+4,0:
POKE C+6,16:POKE C+8,0:POKE P,0
1103 POKE P+2,0:POKE P+4,0:POKE P+6,0:
POKE P+8,0:FOR II=0 TO 15:READ LA
1104 POKE INTIN+II*2,LA:NEXT:VDISYS(1)
:POKE C,23:POKE C+2,0:POKE C+4,0
1105 POKE C+6,1:POKE C+8,2:POKE C,11:P
OKE C+2,2:POKE C+6,0:POKE C+10,1
1106 COLOR 1,1,1,4,4:POKE P,Y:POKE P+2
,Z:POKE P+4,Y+27:POKE P+6,Z+18
1107 VDISYS(1):NEXT Z:NEXT Y:GOSUB 320
:IF DW=0 THEN GOTO 2120
1108 COLOR 1,1,1,FF1,GFG:POKE P,553
1109 COLOR 1,1,1,FF1,GFG:POKE P+2,95:P
OKE P+4,618:POKE P+6,134:VDISYS(1)
1111 POKE Z9,257:GEMSYS(78)
1112 GEMSYS(79):W=PEEK(G+2):X=PEEK(G+4
):OG=PEEK(G+6)
1113 IF W<552 AND X<38 THEN 80
1114 IF W<530 AND X>37 AND X<383 AND O
G=I THEN 1140
1115 IF OG=I AND W>541 AND W<627 AND X
>147 AND X<247 THEN 1120
1116 I=I+1:IF W<530 AND X>37 AND X<383 T
HEN POKE Z9,5:GEMSYS(78):GOTO 1112
1118 POKE Z9,3:GEMSYS(78):GOTO 1112
1120 IF W>601 AND X>227 THEN DW=1:REST
ORE 1188:GOTO 1150
1121 IF W>572 AND X>227 THEN DW=2:REST
ORE 1186:GOTO 1150
1122 IF W>543 AND X>227 THEN DW=3:REST
ORE 1184:GOTO 1150
1123 IF W>601 AND X>207 THEN DW=4:REST
ORE 1182:GOTO 1150
1124 IF W>572 AND X>207 THEN DW=5:REST
ORE 1180:GOTO 1150
1125 IF W>543 AND X>207 THEN DW=6:REST
ORE 1178:GOTO 1150
1126 IF W>601 AND X>187 THEN DW=7:REST
ORE 1176:GOTO 1150
1127 IF W>572 AND X>187 THEN DW=8:REST
ORE 1174:GOTO 1150
1128 IF W>543 AND X>187 THEN DW=9:REST
ORE 1172:GOTO 1150
1129 IF W>601 AND X>167 THEN DW=10:RES
TORE 1170:GOTO 1150
1130 IF W>572 AND X>167 THEN DW=11:RES
TORE 1168:GOTO 1150
1131 IF W>543 AND X>167 THEN DW=12:RES
TORE 1166:GOTO 1150
1132 IF W>601 AND X>147 THEN DW=13:RES
TORE 1164:GOTO 1150
1133 IF W>572 AND X>147 THEN DW=14:RES
TORE 1162:GOTO 1150
1134 IF W>543 AND X>147 THEN DW=15:RES
TORE 1160:GOTO 1150
1135 GOTO 1112
1140 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):ZV=529:POKE Z9,256:GEMSYS(78)
1141 COLOR 1,1,1,1,1:IF S1=W AND S2=X
THEN 1111
1142 POKE P,500:POKE P+2,15:POKE P+4,5
15:POKE P+6,20:VDISYS(1):S1=W:S2=X:I=3
1143 COLOR 1,1,1,FF1,GFG:COLOR 1,1,1,F
F1,GFG:FILL W-1,X-38:LINEF ZV,0,ZV,344
1144 COLOR 0,0,0,0,0:VDISYS(1):GOTO 11
11
1150 GFG=4:POKE C,112:POKE C+2,0:POKE
C+4,0:POKE C+6,16:POKE C+8,0:POKE P,0
1151 POKE P+2,0:POKE P+4,0:POKE P+6,0:
POKE P+8,0:FOR II=0 TO 15:READ LA
1152 POKE INTIN+II*2,LA:NEXT:VDISYS(1)
:POKE C,23:POKE C+2,0:POKE C+4,0
1153 POKE C+6,1:POKE C+8,2:POKE C,11:P
OKE C+2,2:POKE C+6,0:POKE C+10,1
1154 COLOR 1,1,1,4,4:POKE P,553:POKE P
+2,95:POKE P+4,618:POKE P+6,134
1155 VDISYS(1):POKE P,64:POKE P+2,10:P
OKE P+4,111:POKE P+6,25:VDISYS(1)
1156 GOTO 1112
1160 DATA 64639,1344,2336,2720,4752,52
80,9288,10280,18468,20500,36882,40970
1161 DATA 8201,16389,32764,0
1162 DATA 1088,2336,4752,9288,18468,36
882,8201,16388,32770,16388,8201,36882
1163 DATA 18468,9288,4752,2336
1164 DATA 0,65534,32770,32770,32770,32
770,32770,32770,32770,32770,32770,32770
1165 DATA 32770,32770,32770,32770,6553
4
1166 DATA 0,8176,4112,8200,8200,16388,
16388,32770,32770,32770,16388,16388
1167 DATA 8200,8200,4112,8176
1168 DATA 10923,13655,10923,10923,1638
3,0,0,16383,10923,13655,10923,13655
1169 DATA 10923,13655,10923,21655
1170 DATA 8195,16383,8195,16383,16383,
0,0,16383,8195,16383,8195,16383,8195
1171 DATA 16383,8195,16383
1172 DATA 0,65535,0,65535,0,65535,0,65
535,0,65535,0,65535,0,65535,0,65535
1174 DATA 21845,21845,21845,21845,2184
5,21845,21845,21845,21845,21845,21845
1175 DATA 21845,21845,21845,21845,2184
5,21845
1176 DATA 44138,22837,45978,26573,5322
2,40947,16377,32764,65534,32764
1177 DATA 16377,40947,53222,26573,4597
8,22837
1178 DATA 65534,32771,1,1,1,1,1,1,1,1,
1,1,1,1,1,32771
1180 DATA 3,3,3,3,3,3,65535,65535,384,
384,384,384,384,384,65535,65535
1182 DATA 43176,30069,43690,21845,3546
6,22359,43690,21845,43176,30069
1183 DATA 43690,21845,35466,22359,4369
0,21845
1184 DATA 128,128,448,448,992,992,2032
,2032,4088,4088,8188,8188,16382,16382
1185 DATA 32767,32767
1186 DATA 15567,14555,14535,12483,8385
,0,0,0,720,720,720,720,1752,1752,3804
1187 DATA 7374
1188 DATA 16375,16135,16135,16263,1626
1,7943,0,0,2046,4095,8191,15367,14343
1189 DATA 15367,16375,16383
1200 POLY:V=0:Z=0:THE=133:HW=534:RESTOR
E 1202:GOSUB 800:GOSUB 320:BG=0:BE=38
1202 DATA 5,326,85,380,1,7,328,87,382,
0,7,346,87,363,0,5,258,75,312,1,7,260
1203 DATA 77,314,0,7,278,77,295,0,5,19
0,67,244,1,7,192,69,246,0,7,210,69,227
1204 DATA 0,22,98,73,117,1,24,180,75,1
19,0,5,139,80,176,1,7,141,82,178,0
1205 BZ=0:GOTOXY 32,7:?"  0":GOTOXY 3
2,6:?" LINE NO":LINEF 540,121,620,121
1206 NC=4:FOR IU=1 TO 9:READ AA,BS:GOT
OXY 32,AA:?BS:NEXT IU:LPL=2
1207 DATA 9,PLOT,10,X 0,11,Y 0,13,DRAW
,14,X 0,15,Y 0,17,MOUSE POS,18,X,19,Y
1208 POKE P,1:POKE P+2,38:HT1=1:HT2=38
1209 POKE C,11:POKE C+2,2:POKE C+6,0:C
OLOR 1,1,1,CC9,CA9:POKE P,560
1210 POKE P+2,102:POKE P+4,607:POKE P+
```

```
6,117:COLOR 1,1,1,CC9,CA9:VDISYS(1)
1211 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1)
1215 POKE Z9,257:GEMSYS(78):COLOR 1,1,
1
1216 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):OG=PEEK(G+6)
1217 IF W<530 AND X>36 AND X<384 AND O
G=2 THEN 1277
1219 IF W<530 AND X>36 AND X<384 AND O
G=1 THEN 1278
1221 IF W<530 AND X>37 AND X<383 THEN
POKE Z9,256:GEMSYS(78):GOTO 1230
1222 IF W>552 AND X<38 THEN POKE I,1:V
DISYS(1):GOTO 80
1223 IF OG=1 THEN 1260
1224 POKE Z9,3:GEMSYS(78):GOTO 1216
1230 POKE I,3:VDISYS(1):COLOR 1,1,1,CC
9,CA9
1232 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):IF PEEK(g+6)>0 THEN 1211
1233 IF NT=W AND XR=X THEN GOTO 1250
1234 LINEF W-1,X-38,Y,Z:IF W>530 THEN
240
1237 IF X<36 THEN 1240
1238 IF X>383 THEN 1240
1239 NT=W:XR=X:LINEF W-1,X-38,Y,Z:GOTO
 1232
1240 LINEF W-1,X-38,Y,Z:GOTO 1211
1250 LINEF W-1,X-38,Y,Z:POKE I,1:VDISY
S(1):COLOR 1,1,1:GOTOXY 32,18
1251 ?"X"W-1" "':GOTOXY 32,19:?"Y"X-38"
 ";
1252 GOTOXY 32,14:?"X"W-BG-1" ":GOTOXY
 32,15:?"Y"X-BE" ":POKE I,3:VDISYS(1)
1253 GEMSYS(79):W=PEEK(g+2):H=PEEK(g+4
):IF PEEK(G+6)>0 THEN 1239
1254 IF NT=V AND XR=H THEN 1253 ELSE 1
239
1255 GOTO 1239
1260 IF W>560 AND W<607 AND X>102 AND
X<117 THEN POKE Z9,256:GOTO 1285
1261 IF W>64 AND W<111 AND X>10 AND X<
25 THEN CC9=FF1:CA9=GFG:GOTO 1265
1262 IF W>264 AND W<311 AND X>10 AND X
<25 THEN CC9=1:CA9=1:GOTO 1265
1263 IF W>364 AND W<411 AND X>10 AND X
<25 THEN CC9=0:CA9=0:GOTO 1265 ELSE 12
24
1265 POKE C,11:POKE C+2,2:POKE C+6,0:C
OLOR 1,1,1,CC9,CA9:POKE P,560
1266 POKE P+2,102:POKE P+4,607:POKE P+
6,117:COLOR 1,1,1,CC9,CA9:VDISYS(1)
1267 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1)
1268 POKE P,HT1:POKE P+2,HT2:POKE P+4,
HT3:POKE P+6,HT4:GOTO 1224
1270 BZ=0:NC=0:GOTOXY 32,7:?"  ";BZ:PO
KE Z9,256:GEMSYS(78)
1272 COLOR 1,1,1:GOTOXY 32,10:?"X"W-1"
 "':GOTOXY 32,11:?"Y"X-38" "
1273 COLOR CC9,CC9,CC9:LINEF W-1,X-38,
W-1,X-38:BG=W-1:BE=X:Y=W-1:Z=X-38
1274 HT1=W:HT2=X:POKE P+NC,Y+1:POKE P+
NC+2,Z+38:NC=NC+4:GOTO 1211
1277 POKE Z9,256:GEMSYS(78):IF BZ>62 T
HEN 1283 ELSE BZ=BZ+1:GOTOXY 32,7
1278 ?"  ";BZ:COLOR CC9,CC9,CC9:LINEF
W-1,X-38,Y,Z:IF BZ=1 THEN HT3=W:HT4=X
1279 COLOR 1,1,1:GOTOXY 32,10:?"X"W-1"
 "':GOTOXY 32,11:?"Y"X-38" "
1280 LINEF W-1,X-38,W-1,X-38:BG=W-1:BE
=X:Y=W-1:Z=X-38:POKE P+NC,Y+1
1281 POKE P+NC+2,Z+38:NC=NC+4:GOTO 121
1
1283 SOUND 1,13,9,4,6:SOUND 1,0:GOTO 1
211
1285 GEMSYS(78):POKE C,9:POKE C+2,BZ+1
:POKE C+6,0:VDISYS(1):GOTO 1211
1300 SAV:POKE C,32:POKE C+2,0:POKE C+6
,1:POKE I,1:VDISYS(1):POKE Z9,256
1301 GEMSYS(78):POKE I,0:VDISYS(1):POK
E C,11:POKE C+2,2:POKE C+6,0
1302 POKE C+10,1:POKE P,500:POKE P+2,1
5:POKE P+4,515:POKE P+6,20:VDISYS(1)
1303 COLOR 1,1,1,4,2:FOR GI=0 TO 50 ST
EP 10:POKE P,584-GI:POKE P+2,100-GI
1304 POKE P+4,586+GI:POKE P+6,346+GI:V
DISYS(1):NEXT GI:GOTOXY 32,1
1305 RESTORE 1306:FOR YU=1 TO 10:READ
AA,BB,CC,DD,EE,FF:COLOR 1,1,1,EE,FF
1306 DATA 534,50,635,395,4,2,534,50,63
4,394,4,2,539,55,608,89,1,1,541,57
1307 DATA 610,91,0,0,541,74,610,91,0,0
,541,140,623,344,1,1,543,142,625,346
1308 DATA 0,0,609,142,625,346,0,0,539,
361,608,378,1,1,541,363,610,380,0,0
1309 POKE P,AA:POKE P+2,BB:POKE P+4,CC
:POKE P+6,DD:VDISYS(1):NEXT YU
1310 POKE P,543:POKE P+4,625:FOR UQ=15
9 TO 330 STEP 17:POKE P+2,UQ
1311 POKE P+6,UQ:VDISYS(1):NEXT:GOTOXY
 32,1:?" LOAD":GOTOXY 32,2:?" SAVE"
1312 GOTOXY 32,19:?" INDEX";:FOR UT=1
TO 12
1313 GOTOXY 32,UT+5:?" PIC";UT:NEXT:GO
SUB 320:GOSUB 1430:GOTO 1348
1317 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1):POKE C,11
1318 POKE C+2,2:POKE C+6,0:POKE C+10,1
:COLOR 1,1,1,1,1:GOSUB 1505
1319 BT1=0:BT2=0:BT3=0:BT4=0:BT5=0:BT6
=0:BT7=0:BT8=0:BT9=0:BTA=0:BTB=0:BTC=0
1320 ON ERROR GOTO 1321:BLOAD "PIC1":G
OTOXY 36,6:?"*":BT1=1:GOTO 1322
1321 RESUME 1322
1322 ON ERROR GOTO 1323:BLOAD "PIC2":G
OTOXY 36,7:?"*":BT2=1:GOTO 1324
1323 RESUME 1324
1324 ON ERROR GOTO 1325:BLOAD "PIC3":G
OTOXY 36,8:?"*":BT3=1:GOTO 1326
1325 RESUME 1326
1326 ON ERROR GOTO 1327:BLOAD "PIC4":G
OTOXY 36,9:?"*":BT4=1:GOTO 1328
1327 RESUME 1328
1328 ON ERROR GOTO 1329:BLOAD "PIC5":G
OTOXY 36,10:?"*":BT5=1:GOTO 1330
1329 RESUME 1330
1330 ON ERROR GOTO 1331:BLOAD "PIC6":G
OTOXY 36,11:?"*":BT6=1:GOTO 1332
1331 RESUME 1332
1332 ON ERROR GOTO 1333:BLOAD "PIC7":G
OTOXY 36,12:?"*":BT7=1:GOTO 1334
1333 RESUME 1334
1334 ON ERROR GOTO 1335:BLOAD "PIC8":G
OTOXY 36,13:?"*":BT8=1:GOTO 1336
1335 RESUME 1336
1336 ON ERROR GOTO 1337:BLOAD "PIC9":G
OTOXY 36,14:?"*":BT9=1:GOTO 1338
1337 RESUME 1338
1338 ON ERROR GOTO 1339:BLOAD "PIC10":
GOTOXY 36,15:?"*":BTA=1:GOTO 1340
1339 RESUME 1340
1340 ON ERROR GOTO 1341:BLOAD "PIC11":
GOTOXY 36,16:?"*":BTB=1:GOTO 1342
1341 RESUME 1342
1342 ON ERROR GOTO 1343:BLOAD "PIC12":
GOTOXY 36,17:?"*":BTC=1:GOTO 1344
1343 RESUME 1344
1344 POKE 29,3:GEMSYS(78)
1348 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11
1349 POKE C+2,2:POKE C+6,0:POKE C+10,1
:COLOR 1,1,1,1,1
1350 POKE 29,257:GEMSYS(78)
1351 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
```

```
)
1353 IF W)541 AND W<610 AND X)57 AND X
<74 THEN X1=57:GOTO 1365
1354 IF W)541 AND W<610 AND X)74 AND X
<91 THEN X1=74:GOTO 1365
1356 IF W)541 AND W<610 AND X)363 AND
X<380 THEN X1=363:GOTO 1365
1357 IF W)552 AND X<38 THEN 1360 ELSE
1351
1360 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE 1,1:VDISYS(1):GOTO 80
1365 POKE Z9,256:GEMSYS(78):POKE P,541
:POKE P+2,X1:POKE P+6,X1+17
1366 POKE P+4,610:VDISYS(1):LINEF 540,
X1-38,540,X1-38:POKE Z9,257:GEMSYS(78)
1370 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):IF PEEK(G+6)=1 THEN 1380
1371 IF PEEK(G+6)=2 AND X)363 AND X<38
0 AND W)541 THEN 1399
1372 IF X>X1 AND X<X1+18 AND W)540 AND
 W<610 THEN GOTO 1370
1373 POKE Z9,256:GEMSYS(78):VDISYS(1):
LINEF 540,X1-38,540,X1-38:GOTO 1350
1380 SOUND 1,15,5,3,15:SOUND 1,0:IF X1
=57 THEN LOP=1:GOTO 1451
1381 IF X1=363 THEN 1388
1382 IF X1=74 THEN LOP=2:GOTO 1451 ELS
E 1365
1388 POKE Z9,256:GEMSYS(78):VDISYS(1):
LINEF 540,X1-38,540,X1-38:GOTO 1317
1399 POKE Z9,256:GEMSYS(78):LINEF 540,
X1-38,540,X1-38:VDISYS(1)
1400 ON ERROR GOTO 1401:BLOAD "PIC1":B
LOAD "PIC1",MEM1:BT1=1:GOTO 1402
1401 RESUME 1402
1402 ON ERROR GOTO 1403:BLOAD "PIC2":B
LOAD "PIC2",MEM1:BT2=1:GOTO 1404
1403 RESUME 1404
1404 ON ERROR GOTO 1405:BLOAD "PIC3":B
LOAD "PIC3",MEM1:BT3=1:GOTO 1406
1405 RESUME 1406
1406 ON ERROR GOTO 1407:BLOAD "PIC4":B
LOAD "PIC4",MEM1:BT4=1:GOTO 1408
1407 RESUME 1408
1408 ON ERROR GOTO 1409:BLOAD "PIC5":B
LOAD "PIC5",MEM1:BT5=1:GOTO 1410
1409 RESUME 1410
1410 ON ERROR GOTO 1411:BLOAD "PIC6":B
LOAD "PIC6",MEM1:BT6=1:GOTO 1412
1411 RESUME 1412
1412 ON ERROR GOTO 1413:BLOAD "PIC7":B
LOAD "PIC7",MEM1:BT7=1:GOTO 1414
1413 RESUME 1414
1414 ON ERROR GOTO 1415:BLOAD "PIC8":B
LOAD "PIC8",MEM1:BT8=1:GOTO 1416
1415 RESUME 1416
1416 ON ERROR GOTO 1417:BLOAD "PIC9":B
LOAD "PIC9",MEM1:BT9=1:GOTO 1418
1417 RESUME 1418
1418 ON ERROR GOTO 1419:BLOAD "PIC10":
BLOAD "PIC10",MEM1:BTA=1:GOTO 1420
1419 RESUME 1420
1420 ON ERROR GOTO 1421:BLOAD "PIC11":
BLOAD "PIC11",MEM1:BTB=1:GOTO 1422
1421 RESUME 1422
1422 ON ERROR GOTO 1423:BLOAD "PIC12":
BLOAD "PIC12",MEM1:BTC=1:GOTO 1425
1423 RESUME 1425
1425 POKE Z9,3:GEMSYS(78):GOSUB 1505:G
OSUB 1430:GOTO 1350
1430 IF BT1=1 THEN GOTOXY 36,6:?"*"
1431 IF BT2=1 THEN GOTOXY 36,7:?"*"
1432 IF BT3=1 THEN GOTOXY 36,8:?"*"
1433 IF BT4=1 THEN GOTOXY 36,9:?"*"
1434 IF BT5=1 THEN GOTOXY 36,10:?"*"
1435 IF BT6=1 THEN GOTOXY 36,11:?"*"
1436 IF BT7=1 THEN GOTOXY 36,12:?"*"
1437 IF BT8=1 THEN GOTOXY 36,13:?"*"
1438 IF BT9=1 THEN GOTOXY 36,14:?"*"
1439 IF BTA=1 THEN GOTOXY 36,15:?"*"
1440 IF BTB=1 THEN GOTOXY 36,16:?"*"
1441 IF BTC=1 THEN GOTOXY 36,17:?"*"
1442 RETURN
1448 POKE P,541:POKE P+2,X1:POKE P+6,X
1+17:POKE P+4,610:VDISYS(1)
1449 LINEF 540,X1-38,540,X1-38
1450 POKE Z9,257:GEMSYS(78)
1451 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
)
1452 IF W)552 AND X<38 THEN 1360
1453 IF PEEK(G+6)=2 THEN POKE Z9,256:G
EMSYS(78):GOTO 1471
1454 IF W)542 AND W<610 AND X)142 AND
X<346 THEN 1455 ELSE 1451
1455 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
)
1456 IF X)329 THEN X5=329:GOTO 1475
1457 IF X)312 THEN X5=312:GOTO 1475
1458 IF X)295 THEN X5=295:GOTO 1475
1459 IF X)278 THEN X5=278:GOTO 1475
1460 IF X)261 THEN X5=261:GOTO 1475
1461 IF X)244 THEN X5=244:GOTO 1475
1462 IF X)227 THEN X5=227:GOTO 1475
1463 IF X)210 THEN X5=210:GOTO 1475
1464 IF X)193 THEN X5=193:GOTO 1475
1465 IF X)176 THEN X5=176:GOTO 1475
1466 IF X)159 THEN X5=159:GOTO 1475
1467 X5=142:GOTO 1475
1470 POKE Z9,256:GEMSYS(78):VDISYS(1):
LINEF 542,X5-38,542,X5-38
1471 IF LOP=1 THEN IP=57
1472 IF LOP=2 THEN IP=74
1473 LINEF 540,IP-38,540,IP-38:POKE P,
541:POKE P+2,IP:POKE P+4,610
1474 POKE P+6,IP+17:VDISYS(1):POKE Z9,
257:GEMSYS(78):GOTO 1351
1475 POKE Z9,256:GEMSYS(78):POKE P,543
:POKE P+2,X5:POKE P+6,X5+17
1476 POKE P+4,609:VDISYS(1):LINEF 542,
X5-38,542,X5-38:POKE Z9,257:GEMSYS(78)
1477 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):IF PEEK(G+6)=1 AND LOP=1 THEN 1510
1478 IF PEEK(G+6)=1 AND LOP=2 THEN 148
5
1479 IF PEEK(G+6)=2 AND X)363 AND X<38
0 AND W)541 THEN 1478
1480 IF X>X5 AND X<X5+18 AND W)540 AND
 W<629 THEN GOTO 1477
1481 POKE Z9,256:GEMSYS(78):VDISYS(1):
LINEF 542,X5-38,542,X5-38:GOTO 1450
1485 POKE Z9,256:GEMSYS(78):GOSUB 1505
:VDISYS(1):LINEF 542,X5-38,542,X5-38
1486 POKE P,541:POKE P+2,X1:POKE P+6,X
1+17:POKE P+4,610:VDISYS(1)
1487 LINEF 540,X1-38,540,X1-38:ON ERRO
R GOTO 1501
1488 IF X5=142 THEN GOTOXY 36,6:?"*":B
SAVE "PIC1",MEM1,27700:BT1=1
1489 IF X5=159 THEN GOTOXY 36,7:?"*":B
SAVE "PIC2",MEM1,27700:BT2=1
1490 IF X5=176 THEN GOTOXY 36,8:?"*":B
SAVE "PIC3",MEM1,27700:BT3=1
1491 IF X5=193 THEN GOTOXY 36,9:?"*":B
SAVE "PIC4",MEM1,27700:BT4=1
1492 IF X5=210 THEN GOTOXY 36,10:?"*":
BSAVE "PIC5",MEM1,27700:BT5=1
1493 IF X5=227 THEN GOTOXY 36,11:?"*":
BSAVE "PIC6",MEM1,27700:BT6=1
1494 IF X5=244 THEN GOTOXY 36,12:?"*":
BSAVE "PIC7",MEM1,27700:BT7=1
1495 IF X5=261 THEN GOTOXY 36,13:?"*":
BSAVE "PIC8",MEM1,27700:BT8=1
1496 IF X5=278 THEN GOTOXY 36,14:?"*":
BSAVE "PIC9",MEM1,27700:BT9=1
1497 IF X5=295 THEN GOTOXY 36,15:?"*":
BSAVE "PIC10",MEM1,27700:BTA=1
```

```
1498 IF X5=312 THEN GOTOXY 36,16:?"*":
BSAVE "PIC11",MEM1,27700:BTB=1
1499 IF X5=329 THEN GOTOXY 36,17:?"*":
BSAVE "PIC12",MEM1,27700:BTC=1
1500 POKE Z9,3:GEMSYS(7B):GOSUB 1505:G
OSUB 1430:GOTO 134B
1501 RESUME 1502
1502 GOSUB 1505:GOSUB 1430:POKE Z9,3:G
EMSYS(7B)
1503 GOSUB 1505:GOSUB 1430:POKE Z9,3:G
EMSYS(7B)
1504 GOTO 1348
1505 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1)
1506 FOR UQ=6 TO 17:GOTOXY 36,UQ:?" ":
NEXT UQ:POKE I,3:VDISYS(1)
1507 POKE C,11:POKE C+2,2:POKE C+6,0:P
OKE C+10,1:COLOR 1,1,1,1:RETURN
1510 POKE Z9,256:GEMSYS(7B)
1512 ON ERROR GOTO 1526:IF X5=142 THEN
 BLOAD "PIC1",MEM1:BT1=1
1513 IF X5=159 THEN BLOAD "PIC2",MEM1:
BT2=1
1514 IF X5=176 THEN BLOAD "PIC3",MEM1:
BT3=1
1515 IF X5=193 THEN BLOAD "PIC4",MEM1:
BT4=1
1516 IF X5=210 THEN BLOAD "PIC5",MEM1:
BT5=1
1517 IF X5=227 THEN BLOAD "PIC6",MEM1:
BT6=1
151B IF X5=244 THEN BLOAD "PIC7",MEM1:
BT7=1
1519 IF X5=261 THEN BLOAD "PICB",MEM1:
BTB=1
1520 IF X5=278 THEN BLOAD "PIC9",MEM1:
BT9=1
1522 IF X5=295 THEN BLOAD "PIC10",MEM1
:BTA=1
1523 IF X5=312 THEN BLOAD "PIC11",MEM1
:BTB=1
1524 IF X5=329 THEN BLOAD "PIC12",MEM1
:BTC=1
1525 POKE Z9,3:GEMSYS(7B):GOSUB 1505:G
OSUB 1430:GOTO 1448
1526 RESUME 1527
1527 VDISYS(1):LINEF 542,X5-38,542,X5-
38:POKE Z9,3:GEMSYS(7B):GOTO 1450
1600 TEXTX:POKE C,11:POKE C+2,2:POKE C
+6,0:POKE C+10,1:COLOR 1,1,1,1,1
1601 POKE P,544:POKE P+2,87:POKE P+4,6
22:POKE P+6,127:VDISYS(1):POKE P+4,624
1602 COLOR 1,1,1,0,0:POKE P,546:POKE P
+2,B9:POKE P+6,127:VDISYS(1)
1603 POKE P,548:POKE P+2,91:POKE P+4,6
22:POKE P+6,127:VDISYS(1):POKE C,106
1605 POKE C+2,0:POKE C+6,1:POKE I,TE1+
TE2+TE3+TE4+TE5:VDISYS(1):POKE P+2,SZ1
1606 POKE C+2,1:POKE C,12:POKE C+6,0:P
OKE P,0:VDISYS(1):POKE C,0:POKE C+6,3
1607 POKE P,COX:POKE P+2,COY:POKE I,65
:POKE I+2,66:POKE I+4,67:VDISYS(1)
1608 POKE C,12:POKE C+2,1:POKE C+6,0:P
OKE P,0:POKE P+2,13:VDISYS(1)
1610 XCX=CXX:YCY=CYY:POKE C,11:POKE C+
2,2:POKE C+6,0:POKE C+10,1
1611 PS=8:COLOR 1,1,1,1,1:IF Lx=1 THEN
 KOL=1:GOSUB 1929
1620 POKE C,114:POKE C+2,2:POKE C+6,0:
POKE C+10,1:COLOR 1,1,1,4,2:POKE P,535
1621 POKE P+2,130:POKE P+4,633:POKE P+
6,39:VDISYS(1):POKE C,106:POKE C+2,0
1622 POKE C+6,1:POKE I,0:VDISYS(1):POK
E C,11:POKE C+2,2:POKE C+6,0:Xx=0:Px=0
1623 BU=500:RESTORE 1625:FOR IR=1 TO 2
7:READ AA,BB,CC,DD,EE:POKE P,AA+BU
1624 POKE P+2,BB:POKE P+4,CC+BU:POKE P
+6,DD:COLOR 1,1,1,EE,EE:VDISYS(1):NEXT
1625 DATA 37,210,129,358,1,39,212,131,
360,0,41,214,129,35B,0,37,180,57,204,1
1626 DATA 39,182,59,206,0,41,184,57,20
4,0,61,180,B1,204,1,63,182,B3,206,0,65
1627 DATA 1B4,B1,204,0,85,1B0,105,204,
1,B7,1B2,107,206,0,89,1B4,105,204,0
162B DATA 109,1B0,129,204,1,111,1B2,13
1,206,0,113,1B4,129,204,0,37,364,129
1629 DATA 3B6,1,39,366,131,38B,0,41,36
B,129,3B6,0,37,136,57,160,1,39,138,59
1630 DATA 162,0,41,140,57,160,0,61,136
,B1,160,1,63,13B,B3,162,0,65,140,B1
1631 DATA 160,0,B5,136,129,160,1,87,13
B,131,162,0,89,140,129,160,0
1632 COLOR 1,1,1,4,2:POKE P,3:POKE P+2
,35:POKE P+4,52B:POKE P+6,31:VDISYS(1)
1636 O7=0:POKE P+2,2:POKE P+6,35B:FO
R UQ=541 TO 630 STEP 11:POKE P,UQ
1637 POKE P+4,UQ:VDISYS(1):NEXT:POKE P
,541:POKE P+4,62B
1638 FOR UQ=214 TO 360 STEP 18:POKE P+
2,UQ:POKE P+6,UQ:VDISYS(1):NEXT
1639 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11:POKE C+2,2
1670 POKE C+6,0:POKE C+10,1:COLOR 1,1,
1,1,1:POKE P,541:POKE P+4,557:UPx=0
1671 POKE P+2,1B4:POKE P+6,204:VDISYS(
1):LINEF 540,146,540,146:POKE C,12
1672 POKE C+2,1:POKE C+6,2:POKE I,0:PO
KE P+2,13:VDISYS(1):BK=16:POKE P+2,200
1673 POKE C,B:POKE C+2,1:POKE C+6,1:Zx
=0:FOR OO=545 TO 620 STEP 24:BK=BK+1
1674 POKE I,BK:POKE P,OO:VDISYS(1):NEX
T OO:Ax=541:GOSUB 1920:GOTOXY 35,6:?;0
1676 POKE P+2,157:POKE I,4:POKE P,545:
VDISYS(1):POKE I,3:POKE P,570:VDISYS
1677 POKE C+6,5:POKE P,563:POKE P+2,3B
2:POKE I,B3:POKE I+2,B0:POKE I+4,65
1678 POKE I+6,67:POKE I+B,69:VDISYS(1)
1699 Q1Q=1:POKE P,0:POKE P+6,401:POKE
P+4,0:POKE P+2,401:CXx=0:CYx=435
1700 GOTO 1702
1701 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1):POKE Z9,257:GEMSYS(7
8)
1702 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):OG=PEEK(G+6)
1703 IF OG=2 THEN POKE Z9,256:GEMSYS(7
B):PS=0:GOSUB 1970:GOTO 1800
1704 IF W<530 AND X>37 AND X<3B3 THEN
1710
1705 IF W>552 AND X<38 THEN 1980 ELSE
1702
1710 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11:POKE C+2,2
1711 POKE C+6,0:POKE Z9,256:GEMSYS(7B)
1712 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):OG=PEEK(G+6)
1713 IF W>529-SXx THEN W=529-SXx
1714 IF X<39+SYx THEN X=39+SYx
1715 VDISYS(1):POKE P,W+1:IF X>3B2 THE
N X=382
1716 POKE P+6,X:POKE P+4,W+SXx:POKE P+
2,X-SYx-1:VDISYS(1):CXx=W:CYx=X
1717 IF OG=1 THEN POKE Z9,257:GEMSYS(7
8):POKE C,0:GOTO 1715 ELSE GOTO 1701
1719 LINEF W,X-39-SYx,W,X-39-SYx:FOR U
Q=0 TO 127 STEP 2:POKE I+UQ,0:VDISYS
1720 NEXT:CJx=528-W:CIx=SXx-QMx:YC=CJx
/CIx-QNx:Xx=0:Px=0:POKE C,32:TOT=SXx-2
1721 POKE C+2,0:POKE C+6,1:POKE I,1:VD
ISYS(1):POKE C,11:IF YC>64 THEN YC=64
1722 POKE C+2,2:POKE C+6,0:POKE C+10,1
:COLOR 1,1,1,1,1:POKE P,7:POKE P+2,7
1723 POKE P+6,26:POKE P+4,YC*8+10:VDIS
YS(1):COLOR 1,1,1,0,0:POKE P+4,YC*8+12
1724 POKE P,9:POKE P+2,9:POKE P+6,28:V
DISYS(1):COLOR 1,1,1,1,1:GOTOXY 35,6
```

```
1725 ?;YC-PX/2:POKE P,11:POKE P+2,11:P
OKE P+6,26:POKE P+4,18:VDISYS(1)
1730 GEMSYS(79):W=PEEK(g+2)+X:PEEK(g+4
):OG=PEEK(G+6)
1731 IF OX=1 AND OG=1 AND W>537 AND W<
629 AND X>180 AND X<204 THEN 1910
1732 IF OX=1 AND OG=1 THEN 1736
1733 IF W>552 AND X<38 THEN PS=1:GOTO
1980
1734 IF OG=2 THEN POKE Z9,256:GEMSYS(7
8):PS=1:GOSUB 1970:GOTO 1800
1735 OX=1:GOTO 1730
1736 IF W>541 AND W<629 AND X>214 AND
X<358 THEN 1745
1737 IF W>541 AND W<557 AND X>140 AND
X<160 THEN 1788
1738 IF W>565 AND W<581 AND X>140 AND
X<160 THEN 1781
1739 IF Px<2 AND W>541 AND W<629 AND X
>363 AND X<386 THEN 1755
1740 IF W>541 AND W<629 AND X>363 AND
X<386 THEN 1754
1741 IF W>9 AND W<522 AND X>9 AND X<26
 THEN 1940
1742 IF W>548 AND W<622 AND X>91 AND X
<127 THEN PS=1:Zx=0:VPx=0:GOTO 1770
1743 GOTO 1730
1745 GEMSYS(79):W=PEEK(g+2)-541:X=PEEK
(g+4)
1746 IF X>340 THEN Cx=56:GOTO 1756
1747 IF X>322 THEN Cx=48:GOTO 1756
1748 IF X>304 THEN Cx=40:GOTO 1756
1749 IF X>286 THEN Cx=32:GOTO 1756
1750 IF X>268 THEN Cx=24:GOTO 1756
1751 IF X>250 THEN Cx=16:GOTO 1756
1752 IF X>232 THEN Cx=8:GOTO 1756
1753 Cx=0:GOTO 1756
1754 Qx=0:GOTO 1760
1755 Qx=0:Zx=0:GOTO 1761
1756 Dx=W/11:Qx=Cx+Dx+07-64
1760 IF Px<1 THEN Zx=Cx+Dx+07-64
1761 IF YC-Px/2<1 THEN 1730
1762 POKE C,8:POKE C+2,1:POKE C+6,Px/2
+1:POKE P,11:POKE P+2,24
1763 POKE I,Zx:POKE I+Px,Qx
1764 Px=Px+2:VDISYS(1):GOTOXY 35,6:?;V
C-Px/2:Q=0:WAVE 1,5,3,500,0
1765 IF VC-Px/2<1 THEN GOTO 1730
1766 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11:POKE C+2,2
1767 POKE C+6,0:POKE P,Px*4+10:POKE P+
2,10:POKE P+4,Px*4+10:POKE P+6,10
1768 COLOR 1,1,1,1,1:VDISYS(1):POKE P+
4,Px*4+19:POKE P+6,27:VDISYS(1)
1769 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1):VPx=Px:GOTO 1730
1770 GOSUB 1970:PS=0:COLOR 1,1,1,1,4,2:G
OTOXY 35,6:?;0:POKE C,11:POKE C+2,2
1771 POKE C+6,0:POKE C+10,1:POKE P,3:P
OKE P+2,3:POKE P+4,528:POKE P+6,31
1772 VDISYS(1):COLOR 1,1,1,1,1:GOTO 16
99
1780 IF Px/2=0 THEN 1730 ELSE Px=Px-2:
LBx=0:GOTO 1783
1781 IF YC-Px/2<2 THEN 1730 ELSE Px=Px
+2:LBx=1
1783 GOTOXY 35,6:?;YC-Px/2:Ox=0:WAVE 1
,5,3,500,0:POKE C,32:POKE C+2,0
1784 POKE C+6,1:POKE I,3:VDISYS(1):POK
E C,11:POKE C+2,2:POKE C+6,0
1785 IF L8x=0 AND YC-Px/2<2 THEN 1791
1786 POKE P,VPx*4+10:POKE P+2,10:POKE
P+4,VPx*4+10:POKE P+6,10
1787 COLOR 1,1,1,1,1:VDISYS(1):POKE P+
4,VPx*4+19:POKE P+6,27:VDISYS(1)
1791 COLOR 1,1,1,1,1:POKE P,Px*4+10:PO
KE P+2,10:POKE P+4,Px*4+10
1792 POKE P+6,10:VDISYS(1):POKE P+4,Px
*4+19:POKE P+6,27:VDISYS(1)
1795 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1):VPx=Px:GOTO 1730
1800 POKE C,114:POKE C+2,2:POKE C+6,0:
COLOR 1,1,1,4,2:POKE P,535
1801 POKE P+2,130:POKE P+4,633:POKE P+
6,393:VDISYS(1):POKE C,11:POKE C+10,1
1802 COLOR 1,1,1,4,2:POKE P,3:POKE P+2
,3:POKE P+4,528:POKE P+6,31:VDISYS(1)
1804 BU=500:RESTORE 1806:FOR IR=1 TO 2
6:READ AA,88,CC,DD,EE:POKE P,AA+BU
1805 POKE P+2,88:POKE P+4,CC+8U:POKE P
+6,DD:COLOR 1,1,1,EE,EE:VDISYS(1):NEXT
1806 DATA 38,254,127,366,1,40,256,129,
368,0,42,258,127,278,0,42,280,127,300
1807 DATA 0,42,302,127,322,0,42,324,12
7,344,0,42,346,127,366,0,67,152,95,178
1808 DATA 1,69,154,97,180,0,71,156,95,
178,0,50,187,78,213,1,52,189,80,215,0
1809 DATA 54,191,78,213,0,84,187,112,2
13,1,86,189,114,215,0,88,191,112,213,0
1810 DATA 50,221,78,247,1,52,223,80,24
9,0,54,225,78,247,0,84,221,112,247,1
1811 DATA 86,223,114,249,0,88,225,112,
247,0,38,372,127,387,1,40,374,129,389
1812 DATA 0,42,376,85,387,1,87,376,127
,387,0
1815 BP=0:RESTORE 1820:FOR WQ=1 TO 5:R
EAD GT,TG:BP=BP+22:POKE C,106
1816 POKE C+2,0:POKE C+6,1:POKE I,GT:V
DISYS(1):POKE C,8:POKE C+2,1
1817 POKE C+6,TG:POKE P,546:POKE P+2,2
51+8P:FOR UQ=0 TO TG*2-1 STEP 2
1818 READ AP:POKE I+UQ,AP:NEXT UQ:VDIS
YS(1):NEXT WQ:POKE C,106:POKE C+2,0
1820 DATA 1,4,66,79,76,68,2,4,71,82,69
,89,4,4,83,75,69,87,8,9,85,78,68,69
1821 DATA 82,76,73,78,69,69,16,7,79,85,84
,76,73,78,69
1822 POKE C+6,1:POKE I,0:VDISYS(1):POK
E C,12:POKE C+2,1:POKE C+6,0:POKE P,0
1823 POKE P+2,1:VDISYS(1):GOTOXY 34,7
:?"1":GOTOXY 33,9:?"2":GOTOXY 35,9
1824 ?"3":GOTOXY 33,11:?"4":GOTOXY 35,
11:?"5":POKE P+2,13:VDISYS(1):XCX=0
1825 YCV=400:IF SZ1=4 THEN GOSUB 1886
1826 IF SZ1=5 THEN GOSUB 1887
1827 IF SZ1=13 THEN GOSUB 1888
1828 IF SZ1=12 THEN GOSUB 1889
1829 IF SZ1=26 THEN GOSUB 1890
1830 IF TE1=1 THEN X1=258:GOSUB 1870
1831 IF TE2=2 THEN X1=280:GOSUB 1870
1832 IF TE3=4 THEN X1=302:GOSUB 1870
1833 IF TE4=8 THEN X1=324:GOSUB 1870
1834 IF TE5=16 THEN X1=346:GOSUB 1870
1839 POKE Z9,257:GEMSYS(78)
1840 GEMSYS(79):W=PEEK(g+2):X=PEEK(g+4
):OG=PEEK(G+6)
1841 IF OG=1 AND W>554 AND W<612 AND X
>156 AND X<247 THEN GOTO 1880
1842 IF OG=1 AND W>542 AND W<627 AND X
>258 AND X<366 THEN 1855
1843 IF OG=1 AND W>542 AND W<627 AND X
>376 AND X<387 THEN 1850
1844 IF OG=2 THEN POKE Z9,256:GEMSYS(7
8):GOTO 1620
1845 IF W>552 AND X<38 THEN 1980 ELSE
1840
1850 IF Lx=1 AND W>584 THEN KOL=0:Lx=0
:GOSUB 1905
1851 IF Lx=0 AND W<584 THEN KOL=1:Lx=1
:GOSUB 1905
1852 GOTO 1840
1855 IF X>346 AND TX0=0 THEN TE5=16:X1
=346:TX0=1:QMx=0:GOSUB 1870
1856 IF X>346 AND TX0=1 THEN TE5=0:X1=
346:TX0=0:QMx=2:GOSUB 1870
1857 IF X>324 AND TX1=0 THEN TE4=8:X1=
```

```
324!TX1=1:GOSUB 1870
1858 IF X>324 AND TX1=1 THEN TE4=0:X1=
324!TX1=0:GOSUB 1870
1859 IF X>302 AND TX2=0 THEN TE3=4:X1=
302!TX2=1:QNx=1:GOSUB 1870
1860 IF X>302 AND TX2=1 THEN TE3=0:X1=
302!TX2=0:QNx=0:GOSUB 1870
1861 IF X>280 AND TX3=0 THEN TE2=2:X1=
280!TX3=1:GOSUB 1870
1862 IF X>280 AND TX3=1 THEN TE2=0:X1=
280!TX3=0:GOSUB 1870
1863 IF X>258 AND TX4=0 THEN TE1=1:X1=
258!TX4=1:GOSUB 1870
1864 IF X>258 AND TX4=1 THEN TE1=0:X1=
258!TX4=0:GOSUB 1870
1865 GOTO 1840
1870 POKE Z9,256:GEMSYS(78):POKE C,32:
POKE C+2,0:POKE C+6,1:POKE I,3:VDISYS
1871 POKE C,11:POKE C+2,2:POKE C+6,0:P
OKE C+10,1:COLOR 1,1,1,1,1:POKE P,542
1872 POKE P+4,627:POKE P+2,X1:POKE P+6
,X1+20:LINEF 541,X1-38,541,X1-38
1873 VDISYS(1):POKE C,32:POKE C+2,0:PO
KE C+6,1:POKE I,1:VDISYS(1):POKE C,11
1874 POKE C+2,2:POKE C+6,0:POKE C+10,1
:COLOR 1,1,1,0,0:POKE P,548
1875 POKE P+2,91:POKE P+4,622:POKE P+6
,127:VDISYS(1):POKE C,106:POKE C+2,0
1876 POKE C+6,1:POKE I,TE1+TE2+TE3+TE4
+TE5:VDISYS(1):POKE C,8:POKE C+2,1
1877 POKE C+6,3:POKE P,COX:POKE P+2,CO
Y:POKE I,65:POKE I+2,66:POKE I+4,67
1878 VDISYS(1):POKE Z9,257:GEMSYS(78):
X=0:GOTO 1903
1880 IF W)571 AND W<595 AND X)156 AND
X<178 THEN GOSUB 1886
1881 IF W)554 AND W<578 AND X)191 AND
X<213 THEN GOSUB 1887
1882 IF W)588 AND W<612 AND X)191 AND
X<213 THEN GOSUB 1888
1883 IF W)554 AND W<578 AND X)225 AND
X<247 THEN GOSUB 1889
1884 IF W)588 AND W<612 AND X)225 AND
X<247 THEN GOSUB 1890
1885 GOTO 1840
1886 COX=573:COY=111:CXX=571:CYY=156:S
Z1=4:SXx=8:SYx=6:BPx=2:GOTO 1891
1887 COX=570:COY=112:CXX=554:CYY=191:S
Z1=6:SXx=10:SYx=8:BPx=2:GOTO 1891
1888 COX=570:COY=115:CXX=588:CYY=191:S
Z1=13:SXx=10:SYx=16:BPx=3:GOTO 1891
1889 COX=557:COY=115:CXX=554:CYY=225:S
Z1=12:SXx=18:SYx=16:BPx=3:GOTO 1891
1890 COX=553:COY=120:CXX=588:CYY=225:S
Z1=26:SXx=18:SYx=32:BPx=5
1891 POKE Z9,256:GEMSYS(78):POKE C,11:
POKE C+2,2:POKE C+6,0:POKE C+10,1
1892 COLOR 1,1,1,0,0:POKE P,548:POKE P
+2,91:POKE P+4,622:POKE P+6,127:VDISYS
1893 COLOR 1,1,1,1,1:POKE C,32:POKE C+
2,0:POKE C+6,1:POKE I,1:VDISYS(1)
1894 POKE C,11:POKE C+2,2:POKE C+6,0:P
OKE C+10,1:COLOR 1,1,1,1,1:POKE P,CXX
1895 POKE P+4,CXX+24:POKE P+2,CYY:POKE
 P+6,CYY+22:VDISYS(1):POKE P,XCX
1896 LINEF CXX-1,CYY-38,CXX-1,CYY-38:P
OKE P+4,XCX+24:POKE P+2,YCY-1
1897 POKE P+6,YCY+22:LINEF XCX-1,YCY-3
8,XCX-1,YCY-38:VDISYS(1):POKE C,32
1898 POKE C+2,8:POKE C+6,1:POKE I,1:VD
ISYS(1):POKE C,12:POKE C+2,1
1899 POKE C+6,8:POKE P,8:POKE P+2,SZ1:
VDISYS(1):POKE C,8:POKE C+2,1
1900 POKE C+6,3:POKE P,COX:POKE P+2,CO
Y:POKE I,65:POKE I+2,66:POKE I+4,67
1901 VDISYS(1):XCX=CXX:YCY=CYY:POKE Z9
,257:GEMSYS(78)
1903 COLOR 1,1,1,1,1:IF KOL=0 THEN RET
URN
1905 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11:POKE C+2,2
1906 POKE C+6,0:POKE C+10,1:POKE P,548
:POKE P+2,91:POKE P+4,622:POKE P+6,127
1907 VDISYS(1):LINEF 547,91-38,547,91-
38:POKE C,32:POKE C+2,0:POKE C+6,1
1908 POKE I,1:VDISYS(1):RETURN
1910 IF W)689 THEN O7=192:AX=613:GOSUB
 1915:GOTO 1730
1911 IF W)585 THEN O7=128:AX=589:GOSUB
 1915:GOTO 1730
1912 IF W)561 THEN O7=64:AX=565:GOSUB
1915:GOTO 1730
1913 O7=0:AX=541:GOSUB 1915:GOTO 1730
1915 POKE Z9,256:GEMSYS(78):POKE C,32:
POKE C+2,0:POKE C+6,1:POKE I,3:VDISYS
1916 POKE C,11:POKE C+2,2:POKE C+6,0:P
OKE C+10,1:COLOR 1,1,1,1,1
1917 POKE P,AX:POKE P+4,AX+16:POKE P+2
,184:POKE P+6,204:VDISYS(1):POKE P,8X
1918 LINEF AX-1,146,AX-1,146:POKE P+4,
8X+16:LINEF 8X-1,146,8X-1,146:VDISYS
1920 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1)
1921 POKE C,12:POKE C+2,1:POKE C+6,0:P
OKE P,0:POKE P+2,13:VDISYS(1):BX=AX
1922 POKE C,8:POKE C+6,1:FOR VT=229 TO
 355 STEP 18:POKE P+2,VT
1923 FOR O9=543 TO 628 STEP 11:POKE P,
O9:POKE I,O7:O7=O7+1:VDISYS(1):NEXT O9
1924 NEXT VT:POKE Z9,257:GEMSYS(78):RE
TURN
1929 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11:POKE C+2,2
1932 POKE C+6,0:POKE C+10,1:POKE P,548
:POKE P+2,91:POKE P+4,622:POKE P+6,127
1933 VDISYS(1):LINEF 547,91-38,547,91-
38:POKE C,32:POKE C+2,0:POKE C+6,1
1934 POKE I,1:VDISYS(1):RETURN
1940 IF Q1Q=0 THEN 1945
1941 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11:POKE C+2,2
1942 POKE C+6,0:POKE C+10,1:POKE P,CXx
+1:POKE P+2,CYx-SYx-1:POKE P+4,CXx+SXx
1943 POKE P+6,CYx:VDISYS(1):LINEF CXx,
CYx-39-SYx,CXx,CYx-39-SYx
1944 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,1:VDISYS(1)
1945 Q1Q=0:Ox=0:IF Lx=1 THEN 1950
1946 POKE C,106:POKE C+2,0:POKE C+6,1:
POKE I,TE1+TE2+TE3+TE4+TE5:VDISYS(1)
1947 POKE P+2,SZ1:POKE C+2,1:POKE C,12
:POKE C+6,8:POKE P,0:VDISYS(1)
1948 POKE C,8:POKE I,Zx:POKE C+6,Px/2:
POKE P,CXx+4-QMx+QNx:POKE P+2,CYx-BPx
1949 VDISYS(1):GOTO 1968
1950 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11:POKE C+2,2
1951 POKE C+6,0:POKE C+10,1:POKE P,CXx
+1:POKE P+2,CYx-SYx-3:POKE P+4,530
1952 POKE P+6,CYx+2:VDISYS(1):LINEF CX
x,CYx-41-SYx,CXx,CYx-41-SYx:POKE C,32
1953 POKE C+2,0:POKE C+6,1:POKE I,1:VD
ISYS(1):POKE C,106:POKE C+2,0
1954 POKE C+6,1:POKE I,TE1+TE2+TE3+TE4
+TE5:VDISYS(1):POKE P+2,SZ1:POKE C+2,1
1955 POKE C,12:POKE C+6,0:POKE P,8:VDI
SYS(1):POKE C,8:POKE I,Zx
1956 POKE C+6,Px/2:POKE P,CXx+4-QMx+QN
x:POKE P+2,CYx-BPx:VDISYS(1):POKE C,32
1957 POKE C+2,0:POKE C+6,1:POKE I,3:VD
ISYS(1):POKE C,11:POKE C+2,2
1958 POKE C+6,0:POKE C+10,1:POKE P,CXx
+1:POKE P+2,CYx-SYx-3:POKE P+4,530
1959 POKE P+6,CYx+2:VDISYS(1):LINEF CX
x,CYx-41-SYx,CXx,CYx-41-SYx:POKE C,32
1960 POKE C+2,0:POKE C+6,1:POKE I,1:VD
```

```
ISYS(1)
1968 POKE C,12:POKE C+2,1:POKE C+6,0:P
OKE P,0:POKE P+2,13:VDISYS(1)
1969 POKE C,106:POKE C+2,0:POKE C+6,1:
POKE I,0:VDISYS(1):GOTO 1730
1970 IF Q1Q=0 THEN RETURN
1971 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE I,3:VDISYS(1):POKE C,11:POKE C+2,2
1972 POKE C+6,0:POKE P,CXx+1:IF PS=1 T
HEN LINEF CXx,CYx-39-SYx,CXx,CYx-39-SY
x
1974 POKE P+6,CYx:POKE P+4,CXx+SXx:POK
E P+2,CYx-SYx-1:VDISYS(1):POKE C,32
1975 POKE C+2,0:POKE C+6,1:POKE I,1:VD
ISYS(1):CXx=CXx-435:CYx=CYx-435:RETURN
1980 GOSUB 1970:POKE C,12:POKE C+2,1:P
OKE C+6,0:POKE P,0:POKE P+2,13:VDISYS
1981 POKE C,106:POKE C+2,0:POKE C+6,1:
POKE I,0:VDISYS(1):POKE C,11
1982 POKE Z9,256:GEMSYS(78):POKE C+2,2
:POKE C+6,0:POKE C+10,1
1983 RESTORE 1986:FOR UV=1 TO 17:READ
AA,BB,CC,DD,EE,FF:COLOR 1,1,1,EE,FF
1985 POKE P,AA:POKE P+2,BB:POKE P+4,CC
:POKE P+6,DD:VDISYS(1):NEXT
1986 DATA 3,3,528,31,4,2,448,6,519,25,
1,1,450,8,521,27,0,0,452,10,519,25,0,0
1987 DATA 360,6,411,25,1,1,362,8,413,2
7,0,0,364,10,411,25,0,0,260,6,311,25,1
1988 DATA 1,262,8,313,27,0,0,264,10,31
1,25,1,1,160,6,211,25,1,1,162,8,213,27
1989 DATA 0,0,187,10,211,25,1,1,164,10
,187,25,0,0,0,111,25,1,1,62,8,113
1990 DATA 27,0,0,64,10,111,25,0,0
1991 COLOR 1,1,1,FF1,GFG:COLOR 1,1,1,F
F1,GFG:VDISYS(1):POKE C,8:POKE C+2,1
1992 POKE C+6,4:POKE I,66:POKE I+2,85:
POKE I+4,83:POKE I+6,89:POKE P,456
1993 POKE P+2,23:VDISYS(1):POKE C,11:P
OKE C+2,2:POKE C+6,0:POKE P,455
1994 POKE P+2,10:POKE P+4,500:POKE P+6
,10:VDISYS(1):POKE P+2,25:POKE P+6,25
1995 VDISYS(1):POKE Z9,257:GEMSYS(78):
GOTO 80
2120 IF DW=1 THEN RESTORE 1188:GOTO 21
50
2122 IF DW=2 THEN RESTORE 1186:GOTO 21
50
2123 IF DW=3 THEN RESTORE 1184:GOTO 21
50
2123 IF DW=4 THEN RESTORE 1182:GOTO 21
50
2124 IF DW=5 THEN RESTORE 1180:GOTO 21
50
2125 IF DW=6 THEN RESTORE 1178:GOTO 21
50
2126 IF DW=7 THEN RESTORE 1176:GOTO 21
50
2127 IF DW=8 THEN RESTORE 1174:GOTO 21
50
2128 IF DW=9 THEN RESTORE 1172:GOTO 21
50
2129 IF DW=10 THEN RESTORE 1170:GOTO 2
150
2130 IF DW=11 THEN RESTORE 1168:GOTO 2
150
2131 IF DW=12 THEN RESTORE 1166:GOTO 2
150
2132 IF DW=13 THEN RESTORE 1164:GOTO 2
150
2133 IF DW=14 THEN RESTORE 1162:GOTO 2
150
2134 IF DW=15 THEN RESTORE 1160:GOTO 2
150
2135 POKE Z9,257:GEMSYS(78):GOTO 1112
2150 GFG=4:POKE C,112:POKE C+2,0:POKE
C+4,0:POKE C+6,16:POKE C+8,0:POKE P,0
2151 POKE P+2,0:POKE P+4,0:POKE P+6,0:
POKE P+8,0:FOR II=0 TO 15:READ LA
2152 POKE INTIN+II*2,LA:NEXT:VDISYS(1)
:POKE C,23:POKE C+2,0:POKE C+4,0
2153 POKE C+6,1:POKE C+8,2:POKE C,11:P
OKE C+2,2:POKE C+6,0:POKE C+10,1
2154 COLOR 1,1,1,4,4:POKE P,553:POKE P
+2,95:POKE P+4,619:POKE P+6,134
2155 VDISYS(1):POKE P,64:POKE P+2,10:P
OKE P+4,111:POKE P+6,25:VDISYS(1)
2156 POKE Z9,257:GEMSYS(78):GOTO 1112
2500 ALERT:POKE Z9,256:GEMSYS(78):A#=G
B:GINT=PEEK(A#+8):GINN=PEEK(A#+12)
2502 ADDR=PEEK(A#+16):SQP$="[3[ ARE Y
OU SURE !!!][ YES ][ NO "
2503 SQP=varptr(SQP$):SQP1=INT(SQP/655
36):SQP2=SQP-(SQP1*65536)
2504 POKE ADDR,SQP1:POKE ADDR+2,SQP2:P
OKE GINT,1:GEMSYS(52):HQZ=PEEK(GINN)
2505 POKE Z9,3:GEMSYS(78):IF HQZ=2 THE
N POKE Z9,257:GEMSYS(78):GOTO 301
2506 IF ZIP1=1 THEN POKE INTIN,0:VDISY
S(1):GOTO CLEARX
2508 POKE C,32:POKE C+2,0:POKE C+6,1:P
OKE INTIN,0:VDISYS(1):POKE GEM,0:STOP
```

Listing 1: Checksums

```
 1 data  693, 89, 823, 651, 623, 41, 7
05, 635, 712, 257, 5229
 11 data  17, 763, 527, 472, 586, 711,
 291, 370, 374, 487, 4598
 21 data  406, 312, 716, 206, 760, 944
, 236, 340, 446, 640, 5006
 45 data  385, 841, 76, 945, 719, 161,
 691, 46, 471, 100, 4435
 62 data  350, 723, 606, 792, 153, 314
, 752, 154, 678, 563, 5085
 84 data  224, 488, 499, 264, 769, 543
, 683, 707, 327, 98, 4602
 120 data  524, 277, 265, 689, 862, 69
6, 14, 9, 4, 455, 3795
 204 data  36, 31, 12, 7, 2, 451, 28,
23, 18, 999, 1607
 215 data  994, 989, 686, 192, 508, 34
2, 639, 768, 116, 486, 5720
 302 data  806, 608, 941, 0, 317, 785,
 317, 95, 271, 734, 4874
 312 data  293, 560, 678, 269, 89, 302
, 493, 492, 385, 860, 4421
 323 data  506, 798, 605, 571, 57, 984
, 608, 350, 256, 341, 4556
 336 data  523, 766, 587, 819, 817, 34
8, 440, 904, 236, 465, 5905
 358 data  321, 505, 255, 66, 396, 949
, 91, 471, 210, 268, 3532
 370 data  241, 406, 546, 907, 451, 23
3, 758, 758, 768, 753, 5121
 404 data  766, 880, 896, 912, 872, 76
5, 882, 760, 773, 789, 8295
 420 data  540, 99, 138, 107, 514, 181
, 181, 180, 438, 507, 2885
 442 data  467, 392, 602, 312, 563, 58
4, 672, 302, 28, 298, 4212
 458 data  796, 262, 242, 224, 972, 41
0, 20, 232, 692, 920, 4770
 477 data  917, 919, 915, 807, 799, 79
1, 808, 28, 115, 727, 6826
 500 data  473, 494, 937, 610, 246, 61
3, 152, 614, 703, 604, 5446
 611 data  558, 254, 3, 130, 799, 253,
 233, 988, 233, 951, 4402
 627 data  402, 369, 536, 893, 722, 8,
 136, 652, 344, 10, 4072
 651 data  96, 80, 233, 151, 200, 488,
 405, 305, 864, 601, 3503
 670 data  236, 862, 845, 452, 71, 17,
 998, 242, 869, 840, 5424
 683 data  401, 103, 536, 883, 441, 23
```

```
, 999, 832, 14, 347, 4579
 706 data  457, 202, 255, 235, 399, 23
6, 955, 450, 459, 460, 4108
 730 data  217, 352, 462, 285, 420, 18
5, 839, 544, 292, 960, 4556
 741 data  575, 821, 58, 540, 672, 542
, 604, 21, 6, 538, 4377
 758 data  272, 252, 10, 260, 982, 414
, 403, 224, 507, 503, 3827
 778 data  405, 683, 557, 26, 11, 448,
80, 873, 854, 531, 4462
 788 data  533, 509, 190, 176, 299, 71
, 303, 872, 766, 555, 4274
 801 data  936, 634, 692, 132, 729, 70
1, 554, 336, 90, 79, 4883
 814 data  357, 55, 847, 544, 256, 143
, 262, 262, 241, 526, 4596
 830 data  525, 445, 406, 374, 234, 37
8, 553, 576, 542, 221, 4254
 846 data  404, 850, 72, 923, 518, 242
, 8, 310, 303, 400, 4030
 870 data  686, 878, 277, 278, 527, 67
, 598, 492, 528, 255, 4586
 911 data  319, 878, 261, 243, 519, 52
6, 446, 405, 353, 232, 4182
 927 data  377, 447, 573, 722, 252, 47
4, 521, 647, 327, 321, 4661
 943 data  388, 550, 594, 316, 598, 94
6, 639, 603, 270, 256, 5170
 962 data  954, 272, 258, 539, 546, 46
6, 420, 378, 262, 392, 4491
 978 data  467, 588, 742, 268, 9, 83,
922, 534, 155, 543, 4311
 989 data  461, 347, 341, 231, 531, 72
4, 495, 292, 543, 660, 4625
1007 data  434, 749, 191, 519, 406, 7
01, 465, 821, 819, 972, 6877
1021 data  647, 281, 349, 665, 697, 3
07, 292, 930, 92, 747, 5007
1051 data  987, 13, 713, 206, 692, 77
5, 88, 542, 619, 112, 4747
1106 data  149, 587, 103, 619, 295, 4
60, 285, 882, 952, 705, 5147
1118 data  291, 744, 749, 747, 737, 7
42, 759, 768, 773, 771, 7081
1129 data  720, 733, 730, 719, 723, 7
20, 566, 690, 339, 246, 6186
1143 data  893, 649, 852, 543, 620, 1
13, 886, 686, 578, 646, 6458
1161 data  926, 738, 147, 289, 62, 40
4, 48, 400, 408, 643, 4065
1171 data  793, 392, 71, 641, 122, 67
7, 303, 20, 100, 670, 3789
1184 data  690, 230, 574, 568, 600, 8
28, 300, 543, 660, 213, 5206
1205 data  456, 189, 538, 545, 717, 5
62, 588, 707, 471, 841, 5614
1219 data  835, 397, 463, 448, 300, 1
97, 809, 279, 876, 417, 5021
1238 data  457, 473, 794, 58, 560, 94
2, 780, 765, 597, 118, 5544
1261 data  273, 967, 35, 715, 579, 60
5, 999, 953, 484, 599, 6209
1274 data  131, 787, 4, 498, 77, 596,
 743, 433, 609, 68, 3946
1302 data  914, 559, 103, 634, 245, 3
97, 286, 712, 970, 106, 4926
1312 data  260, 328, 863, 542, 141, 2
01, 843, 218, 851, 235, 4482
1325 data  859, 252, 867, 377, 856, 3
51, 845, 372, 853, 393, 6025
1335 data  861, 414, 869, 571, 858, 5
47, 847, 570, 855, 50, 6442
1348 data  870, 133, 303, 803, 309, 2
97, 471, 611, 608, 944, 5349
1366 data  967, 818, 679, 818, 869, 3
04, 603, 127, 890, 566, 6641
1400 data  391, 845, 411, 853, 431, 8
61, 451, 869, 452, 858, 6422
1410 data  425, 847, 445, 855, 465, 8
63, 485, 871, 666, 860, 6782
1420 data  636, 849, 654, 859, 907, 9
48, 954, 960, 966, 999, 8732
1435 data  6, 13, 20, 27, 55, 43, 50,
 457, 71, 314, 1056
1450 data  306, 806, 530, 739, 2, 814
, 44, 16, 40, 50, 3347
1460 data  3, 13, 23, 995, 19, 29, 39
, 340, 645, 753, 2859
1472 data  760, 717, 588, 978, 26, 95
7, 457, 668, 877, 896, 6924
1485 data  4, 71, 777, 674, 704, 687,
 689, 719, 752, 757, 5834
1495 data  762, 795, 5, 990, 22, 903,
 851, 665, 667, 599, 6259
1505 data  604, 140, 430, 303, 625, 9
84, 986, 988, 971, 1, 6032
1518 data  3, 5, 16, 222, 204, 233, 9
17, 875, 619, 210, 3304
1601 data  202, 744, 35, 320, 113, 93
6, 72, 964, 500, 694, 4580
1621 data  821, 70, 820, 304, 554, 57
8, 273, 407, 597, 230, 4654
1631 data  583, 744, 578, 884, 720, 9
1, 454, 258, 4, 778, 5094
1674 data  523, 111, 632, 615, 342, 5
75, 583, 477, 3, 725, 4586
1705 data  618, 74, 632, 478, 632, 52
1, 382, 805, 982, 503, 5627
1720 data  256, 996, 716, 851, 13, 47
4, 476, 922, 440, 835, 5979
1734 data  9, 156, 965, 941, 944, 941
, 968, 525, 782, 585, 6816
1745 data  221, 856, 859, 855, 865, 8
49, 833, 682, 155, 160, 6335
1755 data  738, 987, 555, 887, 331, 2
33, 769, 607, 91, 259, 5457
1768 data  224, 622, 616, 523, 780, 9
43, 42, 117, 917, 141, 4925
1786 data  225, 386, 69, 844, 617, 30
0, 837, 747, 224, 310, 5159
1806 data  531, 613, 579, 604, 435, 5
58, 556, 454, 2, 252, 4584
1818 data  277, 421, 296, 72, 751, 65
4, 590, 404, 466, 466, 4397
1829 data  476, 108, 102, 110, 129, 2
88, 334, 480, 653, 995, 3675
1843 data  992, 59, 537, 651, 652, 59
0, 788, 695, 936, 922, 6922
1859 data  675, 647, 905, 903, 937, 9
37, 597, 330, 671, 55, 6657
1873 data  75, 579, 844, 845, 967, 20
6, 828, 836, 823, 842, 6845
1884 data  801, 599, 771, 796, 73, 72
, 796, 575, 968, 388, 5839
1894 data  853, 887, 315, 16, 924, 19
5, 949, 47, 16, 89, 4291
1906 data  888, 317, 875, 136, 171, 1
8, 298, 337, 312, 871, 4223
1918 data  200, 608, 705, 607, 310, 5
04, 99, 883, 312, 870, 5098
1940 data  541, 85, 124, 655, 618, 77
0, 89, 275, 129, 899, 4185
1950 data  84, 504, 933, 983, 324, 68
7, 144, 926, 918, 947, 6050
1960 data  374, 87, 918, 53, 88, 273,
 986, 78, 65, 876, 3790
1982 data  318, 691, 265, 559, 518, 5
18, 353, 290, 926, 503, 5021
1993 data  252, 945, 558, 851, 852, 8
53, 854, 855, 863, 864, 7747
2127 data  865, 866, 917, 917, 917, 9
17, 917, 917, 546, 854, 8633
2151 data  545, 622, 115, 888, 688, 5
50, 982, 922, 47, 755, 6114
2505 data  829, 972, 398, 2199
```

The Spacer

How to make your hard disk fast again

by Dave Small, Copyright 1987

Well, so you're the proud owner of a hard disk. Welcome to the club! Isn't it great?

You're amazed by the sheer *speed* of the hard disk. Copy a file to it? Zip. Copy a file from the hard disk, to the hard disk? Zip, zip. We're talking performance far higher than an IBM. A 200K file can be copied in three seconds on the ST; it takes ten seconds on a PC.

But something seems to have happened to your hard disk recently. It's—heavens!—bogging down. Slooooww.

In fact, the delays are getting ridiculous. You try to copy a file on the hard disk, and you get a zip sound, then, a slow blink, blink, blink of the drive light, and 15 seconds later, the zip as the operation completes.

Well, you've got a problem. In fact, we've all got it. It's the dreaded FAT Lookup Problem. Let's find out about it, and how to fix it.

SOME TRIVA

Your floppy disk is composed of 720 512-byte sectors. Oddly enough, the sectors are numbered 1-720. (If you go double-sided, make that 1-1440). 720 sectors at 512 bytes per sector is 360,000 bytes per side of the disk; that's where your ST's disk size comes from.

Now, a hard disk is just like a floppy disk in that there are a bunch of 512-byte sectors. However the quantity is much larger. A 5-megabyte hard disk has 10,000 sectors! Your standard Atari SH204, a 20-megabyte unit, has 40,000 sectors on it.

The operating system worries about splitting up these individual sectors into files, subdirectories, and so forth. It keeps a table, called a File Allocation Table, which is a map of all the sectors. It marks each sector as used, unused, or bad.

Unfortunately, the folks who wrote TOS (The Operating System) made a little mistake before committing it irrevocably to ROM: the routine that processes the FAT is quite slow. In specific, scanning a FAT for an "open" sector to write to is *extremely* slow. You do this scan anytime you write something new to the disk; the operating system is finding free sectors to write to.

It was a natural mistake. Working with floppy disks, they never noticed the speed problem. There's only 720 or so sectors on a floppy, and the FAT search problem wasn't noticeable. In fact, until you get over a megabyte of data on the storage media, you don't notice it at all.

However, hard disk owners tend to collect lots of data, and then the problems become extremely evident. Let's go through a sample copy. Assume you have a hard disk partitioned into four 5-megabyte parts, a very common thing to do. You're copying a file from disk C to disk F. Assume also that disk F is rather full.

You drag the icon, and let go. First, you notice the drive light go on and the head move; it's pulling the file off drive C. Then it begins to write.

The Spacer

Slowly, very slowly, it reads in the FAT for the disk F. It's looking for a place to put the data it just read. You'll see the drive light blinking—and anytime the ST goes to slow the hard disk light has time to go out, you're in trouble. Finally, it finds a home for the data. Zip! The data's written.

Of course, it took a while.

If you think I'm kidding, let's look at the times to duplicate a 200K file on the hard disk, only varying how much data is out there first.

Table 1: Disk timings. Duplicating a 200K file.
Disk empty: under 3 sec.
Disk 5 megabytes: 30 sec.
Disk 10 megabytes: 45 sec.
Disk 15 megabytes: 63 sec.

This is *bad*. All that extra time is spent diddling with the FAT.

In contrast, an IBM PC takes around ten seconds to duplicate a 200K file, regardless of how full the disk is; they've got a good fast FAT search algorithm, so the overhead doesn't slow things down. [Why ten] seconds? Mainly, a slower hard disk and slower processor.

If you're using something like **Alcyon C**, which creates six temporary files during the compile/assembly process (No? Look at AS68 closely. It takes three by itself!) you're going to be old by the time it's done. And why was it you bought a hard disk? Speed?

Those of you familiar with my style probably think I have some magic desk accessory or program to fix this. Alas, I don't. The problem is deeply rooted in the slippery, slimy entrails of TOS, and like any digestive system problem, it's hell to diagnose and fix. It's all in ROM, just to make it worse.

I do, however, have a technique to speed your hard disk back up when you have a lot of data out there: It's called "spacing." I believe that all of you suffering from the HD slowdown will benefit a lot from this technique.

Let's look at the FAT process again. The slowdown occurs when the operating system is looking over a huge disk, with thousands of sectors, trying to find the first sectors that are free to be written to. The problem is only in writing, not in reading.

On a hard disk, the first data written to it (the first file copied to it after formatting) is written in the first sectors (e.g., sectors 1-xxx of the 40,000 available). The next file goes in the next sector, and so on.

The farther we get away from the start of the hard disk, the longer the hard-disk driver (FAT) takes to do the write. So, the first conclusion is, keep your disk nearly empty if you want it to stay fast.

This is fine, but there are those of us who want to store lots on our hard disk and still have it move quickly. I mean, heck, the Alcyon C compiler takes about a megabyte, what with all its support files. And what about all those good downloads? NEOchrome files?

For those of you that want lots of data and peppy performance, we have the spacer technique.

Let's imagine how the hard disk looks after you've copied, typically, the Alcyon C compiler to it (which, with all its subfiles, takes up lots of room). Let's further assume a 5-megabyte partition (segment of hard disk).

Sector 1 (start of 5-megabyte area)
Alcyon C and files: comprise about 1-megabyte
Section 2,000
Empty area
Sector 10,000 (end of 5-megabyte area)

Now if you go do a compile, all those temporary files, and all your editing, gets done starting at sector 2,000. Every time you write to the disk, you have to sit and wait while the ST hunts through the first used 2,000 sectors. That will bog you down.

Most people's hard disks look like this. Data gets put on them willy-nilly, whenever it's available, in the first open sector.

Instead, let's do something different. Before copying anything to the hard disk—when it's new, freshly formatted and zeroed, let's put a huge file on the hard disk called a "spacer." One easy way to make a spacer is to create a folder and fill it with long files. I use a listing of the **Magic Sac**, which is about 500K. However, feel free to use any long files that come to mind.

Let's make our spacer 4 megabytes long, and copy it to the hard disk. It looks like this now:

Sector 1 (start of 5-megbyte area)
Spacer folder files (temporary, 4 megabytes)
Sector 8,000
Empty area
Sector 10,000 (end of 5-megabyte area)

Only, now copy the Alcyon files to the hard disk. Of course, this is going to take awhile, because they're so far away from the start of the hard disk, so go get some coffee while it works:

Sector 1 (start of 5-megabyte area)
Spacer files (temporary, 4 megabytes)
Sector 8,000
Alcyon C (1 megabyte or so)
Sector 10,000 (end of 5-megabyte area)

Okay, your hard disk is now full.

Go delete the spacer. The Alycon C files aren't going to move, which is one of the secrets of this technique, so we have:

Sector 1 (start of 5-megabyte area)
Empty space
Sector 8,000
Alcyon C (1 megabyte or so)
Sector 10,000 (end of 5-megabyte area)

Now we're getting somewhere. When the operating system makes a new, temporary file, it doesn't have to search very far: There's a big empty area at the beginning of the disk! In tests I've run, this has increased the performance of compilers by 300%.

When you think about it, how many of the things you put on the hard disk are ever modified? If you're like me, there's piles of utilities and the like that never get changed. With the Spacer technique, you can put them on the end of the hard disk, where they won't be slowing down your work.

The amount to space your files across the disk is strictly up to you. I'd go at least halfway across the hard disk, but it's up to you. Again, the idea is to first use up the "fast" part of the hard disk with a spacer, which forces the following files to be written in the "slow" part. When you get rid of the spacer, you'll do all your work on a remarkably fast hard disk.

Won't your utilities slow down?

No. Remember, the bug is only while writing to the disk, not while reading. All you do with a utility is read it in (like the C compiler code). Temporary files and such are what you're concerned with.

Periodically, as things slow down, you may need to go respacer the hard disk to get your performance back.

C MANSHIP PROGRAMMING

by Clayton Walnum

Everyone who's tired of studying GEM's windows, please raise your hand. Yeah, that's what I thought. Okay, it's time to take up a new subject, something that, though it'll give you a lot of information on how your computer works, won't give you a headache trying to understand it.

One of the more useful things about the ST is the ability to have many screens of data in memory at once and flip between them as you like. I thought this would be a good subject to tackle, since it enables us to not only see how we can accomplish "screen flipping" (which is really a simple process), but how to apply some of the other techniques we've learned, such as the programming of file selector boxes. We'll also take a look at some new information, such as the **DEGAS** picture file format.

Type in this month's listing and compile it. Note that the program was developed using the **Megamax C** compiler. If you have a different compiler, you may need to make some small changes to the code. Once you've got the program compiled and linked, go ahead and run it.

What we're going to do is load two DEGAS format pictures into memory, and then use an alert box to choose which picture to view. We'll have to tell the program which files to load, so the first thing the program will do is bring up a file selector box. Use it in the normal way to select two DEGAS pictures for *loading*.

While you're doing this, keep in mind that the program presented here is a very stripped-down model. In other words, it doesn't incorporate much in the way of error checking. In fact, it'll let you load just about any type of file into memory, whether it's DEGAS or not. So do your own error checking, and make sure you're selecting the right type of file.

If you click on the file-selector box's Cancel button for either picture, or if the program gets a file error, you'll be returned to the desktop.

Once you get two files loaded, an alert box with three buttons will appear. Clicking on the first button will cause the first loaded picture to be displayed. Clicking on the second button will show the second picture. The Quit button should be used to leave the program and return to the desktop. Once a picture is displayed on the screen, clicking the left button will bring the alert box back, allowing you to make another choice or quit the program.

Hey! That space is reserved!

The first step is getting our picture files loaded into the computer is figuring out where we're going to store them. We need a lot of space—32K for each picture—and we have to make sure that, wherever we store the picture information, it doesn't get in the way of our program or its data. Also, since we're going to be displaying a couple of different screens, we have to make sure we store the address of the original screen, as well as its color palette, so that we can restore it when the program's finished.

Keep in mind that the program presented here is a very stripped-down model.

Take a look at the function **init_screens()** in Listing 1. The first thing we do here is store the desktop's color palette with the line:

```
for (x=0; x<16;
desk_palette[x++]
= Setcolor(x,-1));
```

The function **Setcolor()** is an XBIOS function and is defined in the OSBIND.H file. This function requires two integers as arguments. The first is the index of the color you want to change (from *0* to *15*), and the second is the color to change it to.

Colors on the ST are formed by mixing the correct proportions of red, green and blue, each of which can have a value from *0* (minimum) to *7* (maximum). The color value for blue is placed in the first nibble (four bits) of the integer; the value for green is placed in the second nibble; and the value for the red is placed in the third. This works out well in hexadecimal: *0x007* is the brightest blue; *0x070* is the brightest green; and *0x700* is the brightest red. White is all the values at their maximum (*0x777*), while black is formed by setting all colors to the minimum *0x000*). By combining the three basic colors in varying intensities, we can conjure up any of the ST's 512 possible colors.

But all that *is besides* the point (go ahead and boo; I deserve it). We don't want to change the colors (at least, not yet); we want to know what value they're currently set at, so we can store them for later retrieval. One thing I didn't tell you about the **Setcolor()** function is that it always returns a color's previous setting (its color value before we changed it). If we make the second argument a negative number, it won't change the color register at all; it'll just return the color's setting.

Now you can see how the above code segment works. We use a **for** loop to step through all 16 possible elements of the color palette, calling **Setcolor()** in each iteration with a color value of *-1*, in order to have the current color returned to us. Each of these colors is stored in the array **desk_palette[]**, where they'll be when we're ready to restore the desktop's colors.

Now that we've gotten that taken care of, we have to store the address of the desktop's screen (we do want to get back there eventually, you know). This line takes care of that:

```
scrn = Physbase();
```

Here, the variable **scrn** is a long integer that'll hold the address returned from **Physbase()**. The function **Physbase()** returns the address of the physical screen, the area of memory currently displayed on your monitor. The function **Logbase()** returns the

address of the logical screen, an area of memory where all output to the screen is to go.

In most cases, the physical and logical screens are in the same location. For example, as I'm writing this article, I can see the new text I'm typing appearing on the screen. That means that the displayed screen and the one the program is sending text to are at the same address. Sometimes, though, you may find it handy to direct data to a different place in memory, so you can do the screen updating "behind the user's back." Once the logical screen has been set up the way you want it, you can simply flip to it, creating the illusion of the screen being intently updated. We'll see how all this works a little later on.

Now that we know where our physical screen is, we're ready to allocate some memory for a couple of logical screens. You allowed only one physical screen, *but you can have* as many logical screens as you can store in memory. In the function **init_screens()**, we set up a **while** loop that first allocates a block of screen memory, then calls a function to read the picture data into it. To allocate a block of memory, we use the call:

```
addr = Malloc(bytes);
```

Here, the pointer **addr** will hold the address of the block of memory, and the long integer **bytes** is the number of bytes you wish to reserve. This function returns a *0* if the amount of memory you've requested isn't available. One variation on the **Malloc()** call, making **bytes** equal to *-1L*, will return the total amount of memory available.

You've probably noticed, though, that our call to **Malloc()** in Listing 1 looks quite a bit more complex:

```
pic[x] = (Malloc(32768L) &
         0xffffff00 ) + 0x0100;
```

First, even though **pic[x]** doesn't look like a pointer, it is. In fact, **pic[]** is an array of pointers (actually, long integers, but for our use that amounts to the same thing). For programming purposes, it's very convenient to store the addresses of our screens in an array, so that we can get at them easily with some sort of loop.

Next comes that strange looking **Malloc()** call. It looks strange to you because there's one little detail I've yet to mention, the fact that the ST's screen memory must always start on a 256-byte boundary. And, since **Malloc()** doesn't know or care about this little requirement, it's up to us to smooth things over.

The first step in getting to a safe 256-byte boundary is to use C's AND operator to mask off the eight rightmost bits of the address, using the hex value *0xFFFFFF00* as our mask. This value has every bit set except the rightmost eight. The AND operator compares the bits of two values, returning a true *(1)* when both bits are on and a false *(0)* when either or both the bits are off. What that means for us is that every bit we have off in the mask will result in a *0* in the bit it's being ANDed with. Let's say the address returned from **Malloc()** was *0x0034CC3E2*. After ANDing it with our mask, we'd have *0x034CC300*, which is an address on a 256-byte boundary.

But even though we're now on the boundary we wanted, it's not a safe boundary. Why? Because the address we have now is lower than the one returned from **Malloc()**. We're no longer in the area we just reserved; we're actually before it. If we try to load data there, we'll probably end up clomping all over our program—and get a delightful string of bombs up on the screen.

That's why, after completing the AND operation, we add *0x00000100 (256* decimal) to the resultant address. That pushes it back into our reserved area.

"Ah!" you cry in that smug manner you use when you think you've caught the professor with his foot in it. "If we're pushing the address forward, doesn't that mean that, when we load our picture data, the last few bytes will be placed outside the reserved area, beyond the other end?"

Nope. You see, we've reserved 32768 bytes (that's a full 32K), and we only really need 32000 bytes for our picture data. When people tell you that screen memory on the ST is 32K, they're not telling you the whole truth. It's actually a bit short of a full 32K. We just like to round it off when we speak. (You ever hear people refer to the SF314 disk drive as a one-meg drive, even though you can only story 720,000 bytes on the disk? Same idea.)

One thing we do have to watch out for, though, is how we handle any subsequent calls to **Malloc()**, because it doesn't know we've finagled the address it gave us the first time around. The next time we allocate some memory, we have to remember to add the same amount to the returned address, or we're sure to make digital footprints in the previous areas. And digital footprints often result in the Big Kablooey. (In our case, since we're using those areas only for a screen display, we'd simply end up with some funny looking pictures.)

Okay, we've got the memory we need to store our pictures. Now let's think about how we're going to load them. The first step is to get the picture's filename, and the obvious way to do that is with GEM's handy file selector box. Included in Listing 1 is a function called **select_file()**. This is a generic file selector box routine that I came up with that you can use in your own programs. It handles some of the minor details for you, allowing you to just call a file selector box and have the complete filename (including the path) returned to you. (You're welcome.)

If you look at the function **get_pic()**, you'll see how we get started. First, because it's required by **select_file()**, we have to come up with a default filename. This will be tacked on to the end of the pathname field in the file selector box, and allowing us to narrow the number of files shown when the box first comes up. In our example, we start with the string "*.P1" then finish the default name by adding the proper DEGAS resolution indicator. Adding the ASCII value of "1" to the value returned from **Getrez()** performs that trick.

Our file selector function, **select_file()**, returns the complete chosen filename and the button that was clicked to exit the file selector box. The call to the function looks like this:

```
select_file(path,file,
default,flag);
```

Here, **path** is a pointer to a 64-byte character array where the function will store the completed filename. The pointer **file** is the address of a 13-byte character array that'll hold the selected filename after the call to **fsel_input()**. You may also, before the function call, store a filename here that you want to appear in the filename field of the file selector box. The pointer **default** contains the address of a string containing the text you want added to the selector box's pathname field. And finally, **flag** is a Boolean value that tells the function whether you want the string pointed to by **file**

to appear in the file-selector box's file field.

It sounds a little complicated at first, but I've found that using this function is a lot easier than trying to remember how to handle the file-selector box each time I need it.

As I mentioned before, **select_file()** returns the value of the file-selector button that was clicked. Strangely enough (or perhaps it was done purposely), these values also correspond to obvious Boolean values; the Cancel button returns *0*, and the OK button returns *1*. In the function **get_pic()**, we use this returned value as a Boolean to evaluate an **if** statement. In other words, if the user clicks on the file selector box's Cancel button for either of the two files we're going to be loading, we'll know not to read the file and instead exit the program.

If the user clicks the file selector's OK button, we call the function **read_degas()** to attempt to load the file chosen. If the file loads all right, this function will return a value of TRUE. If an error is encountered (maybe the file doesn't exist), it returns a value of FALSE. We use this returned value in another **if** statement to determine whether we should continue or return to the desktop. In a full-scale application program, you would want to give the user a message if you ran into an error, but for the sake of brevity, we've kept things to a minimum in the example program.

Turn your attention now to **read_degas()**. It's here that we actually read the selected picture file into memory. This function needs to know which picture we're loading and the complete filename. The first thing we must do is open the file, but we have to make sure we open it to **read binary**. We covered the **open()** function way back in Issue 4, but we didn't talk about the 0_BINARY flag. When we open the file with this flag (it's defined at the top of the listing as *8192*), we're telling the system that we want the file read from the disk in an untranslated form, as a continuous block of data, rather than a series of lines ending with carriage returns and line feeds.

Before we go any further, we need to discuss the format in which DEGAS pictures (the unsqueezed variety) are saved to disk. If you've ever looked at a disk directory containing these picture files, you've undoubtedly noticed that they are 32034 bytes. In order to get the picture up on the screen properly, we have to know what each of these bytes is.

The first two bytes of a DEGAS file indicate the picture's resolution. It's interpreted as a word value: *0x0000*, *0x0001* or *0x0002*, for low, medium or high resolution, respectively. Normally, we'd want to check the resolution of the picture against the computer's current resolution, to make sure they aren't different, and if they are, give the user an error message. But, as I said before, for the sake of brevity, we're going to do things quick and sloppy and just throw away those two bytes after we've read them.

The next 32 bytes (16 words) are the picture's color palette. That we don't want to throw away; we want to read it into the array we've set up for storing this information.

Finally, the last 32000 bytes are the actual picture data. We read that information into the area of memory starting at the address stored in the appropriate element of the **pic[]** array.

Now that we've got all the data read, we close the file and return a value of TRUE to the calling function. Notice that, in the function **read_degas()**, we're using the value returned from the **open()** function in an **if** statement. Doing this makes sure, in the case of a file error, that we skip over all the subsequent file handling code, and just return from the function a value of FALSE.

Once we get two picture files loaded okay, program execution gets turned over to the function **flip_screens()**, where we get a chance to actually view the pictures. We begin by calling up an alert box with three buttons, one button for each picture plus a Quit button. We use the value returned from the alert box as an index into the **pic[]** array, where the pointers to the screens are stored. To flip between the different screens, we use the call:

```
Setscreen (log,phys,res);
```

Here, **log** is the address of the logical screen, **phys** is the address of the physical screen, and **res** is the screen resolution we want to switch to. If we don't want to switch screen resolutions, we just give **res** a negative value. In fact, all parameters with a negative value will be ignored.

In most cases, you would set both the logical and physical screen to the same address. As for the resolution, you'll almost always want to leave it unchanged (use a negative value) because GEM isn't ever informed of resolution changes, and that little bit of ignorance on its part can lead to some pretty nasty complications.

Exactly flipping the screen, we wait for a mouse button click using a call to **evnt_button()**, after which we bring up the alert box to get another choice. We keep displaying the selected picture until the Quit button is clicked. Then we close things up and return to the desktop.

Putting it back where we found it

But we can't just go blithely on our way, returning to the desktop by just closing the virtual workstation and calling **appl_exit()** as we've gotten used to doing. Nosiree. We've got some cleaning up to do first. We've allocated a bunch of memory for our picture files, and before we leave, we have to get it back. Not a tough thing to do. The following call will return a block of memory (one that was allocated with **Malloc()** to the system:

```
Mfree (adr);
```

The pointer **adr** is the address of the block we want to de-allocate. You need to make a separate call to **Mfree()** for each block allocated, and you must return the blocks in the reverse order you allocated them.

Once we've returned all the memory to the system, we can exit the program in the usual manner. You can see all this being done in Listing 1 in the function **clean_up()**.

The whoops department

I was going over a couple of the old installments of **C-manship** the other day and found that I had left something out of our discussion of resource files. So let's correct that oversight, shall we?

When you load a resource file, it takes up a certain amount of memory, and, just like the picture files we were working with this month, that memory should be returned to the system before we exit our program. The call to free the memory allocated for a resource file is:

```
rsrc_free ();
```

You should use this call (whenever you've loaded a file with **rsrc_load()**) before you exit a program or before you attempt to load a different resource file.

```
                /*************************************************/
                /*            C-manship, Listing 1               */
                /*                 ST-Log #20                    */
                /*          Developed with Megamax C             */
                /*************************************************/



        #include <osbind.h>

        #define TRUE        1
        #define FALSE       0
        #define O_BINARY    8192
        #define QUIT        3
        #define LEFT_BUTTON 1
        #define DOWN        1

        /* The usual required GEM global arrays */
        int work_in[11],
            work_out[57],
            pxyarray[10],
            contrl[12],
            intin[128],
            ptsin[128],
            intout[128],
            ptsout[128];

        /* Global variables */
        int handle, dum;

        long pic[2],  /* Pointers to logical screens. */
             scrn;    /* Pointer to physical screen.  */

        int desk_palette[16];   /* Desktop color palette.  */
        int pic_palette[2][16]; /* Picture color palettes. */


        main ()
        {
           appl_init ();        /* Initialize application.        */
           open_vwork ();       /* Set up workstation.            */
           do_pictures ();      /* Go do the picture stuff.       */
           clean_up ();         /* Get everything back to normal. */
           appl_exit ();        /* Back to the desktop.           */
        }


        open_vwork ()
        {
           int i;

           /* Get graphics handle, initialize the GEM arrays and open  */
           /* a virtual workstation.                                   */

           handle = graf_handle ( &dum, &dum, &dum, &dum);
           for ( i=0; i<10; work_in[i++] = 1 );
           work_in[10] = 2;
           v_opnvwk ( work_in, &handle, work_out );
        }


        do_pictures ()
        {
           /* If the pictures are loaded okay, */
           /* then allow user to view them.    */

           if ( init_screens () )
              flip_screens ();
        }


        init_screens ()
        {
int x,     /* Index variable.              */
    okay;  /* File load flag.              */
```

```
      /* Store the desktop's color palette. */
      for ( x=0; x<16; desk_palette[x++]=Setcolor (x, -1) );

      /* Store the address of the desktop's screen. */
      scrn = Physbase ();

      /* Reserve memory for pictures and load them */
      /* into the allotted space, storing pointers */
      /* to them in the pic[] array.               */

      okay = TRUE;


   x = 0;
   while ( (okay == TRUE) && (x < 2) ) {
      pic[x] = ( Malloc (32768L) & 0xffffff00 ) + 0x0100;
      okay = get_pic ( x++ );
   }
   return ( okay );
}


flip_screens ()
{
   int choice; /* Button number clicked in alert box. */

   choice = 1;

   /* View pictures until QUIT button is clicked. */
   while ( choice != QUIT ) {

      /* Call up alert box to get user's picture choice. */
      choice = form_alert ( 0, "[2][Choose picture to view][One|Two|Quit]" );

      /* We only want to show a picture if the */
      /* QUIT button hasn't been clicked.      */

      if ( choice != QUIT ) {

         /* Set the screen to show the chosen picture. */
         Setscreen ( pic[choice-1], pic[choice-1], -1 );

         /* Set the palette to the picture's settings. */
         Setpallete ( &pic_palette[choice-1][0] );

         /* Wait for a button click. */
         evnt_button ( 1, LEFT_BUTTON, DOWN, &dum, &dum, &dum, &dum );
      }
   }
}


get_pic ( num )
int num; /* Number of picture to load. */
{
   char path[64],    /* Storage for picture's pathname.       */
        file[13],    /* Storage for picture's filename.       */
        pictype[6]; /* Storage for default picture filename. */

   /* Build default picture filename. */
   strcpy ( pictype, "*.PI " );
   pictype[4] = Getrez () + '1';

   /* If file selector CANCEL button wasn't clicked, */
   /* read the chosen DEGAS file into memory.  If an */
   /* error is returned, the program will abort.     */

   if ( select_file ( path, file, pictype, FALSE ) )
      if ( read_degas ( num, path ) )
         return ( TRUE );
      else
         return ( FALSE );
   else
      return ( FALSE );
}


read_degas ( num, pathname )
int num;          /* Picture number to read. */
char *pathname; /* Picture's pathname.     */
```


```
{
   int f_h,      /* File handle.                    */
       buf[10]; /* Temp buffer for unused bytes. */

   /* Process file only if no error is returned when opening. */
   if ( (f_h = open ( pathname, O_BINARY )) != -1 ) {

      /* First two bytes is resolution data. */
      read ( f_h, buf, 2 );

      /* Next 32 bytes (16 words) is the color palette. */
      read ( f_h, &pic_palette[num][0], 32 );

      /* Finally, we have 32K of picture data. */
      read ( f_h, pic[num], 32000 );

      /* Close file and tell calling function */
      /* that everything went all right.      */
      close ( f_h );
      return ( TRUE );
   }

   /* In case of error opening the file. */
   else
      return ( FALSE );
}


select_file ( path, fnme, deflt, display)
char *path,  /* Address for path storage.     */
     *fnme,  /* Address for filename storage. */
     *deflt; /* Address of default filename.  */
int display;    /* Display default filename?     */
{
   int x,       /* Loop variable.                          */
       choice, /* Button clicked from file selector box. */
       len;    /* String length.                          */
   char ch;    /* Temp character storage.                 */

   /* Clear filename string if not to be displayed. */
   if ( display == FALSE )
      for ( x=0; x<13; fnme[x++] = '\0' );

   /* Build file selector box pathname. */
   Dgetpath ( path, 0 );
   len = strlen ( path );
   path[len] = '\\';
   strcpy ( &path[ len + 1 ], deflt );

   /* Call up file selector box to get user's choice. */
   fsel_input ( path, fnme, &choice );

   /* Find last significant character in pathname in    */
   /* order to delete the filename portion of the path. */

   len = strlen ( path );
   x = len-1;
   while ( path[x] != '\\' && path[x] != ':' && x > 0 )
      --x;
   strcpy ( &path[x+1], fnme );

   return ( choice );
}


clean_up ()
{
   /* Setscreen back to desktop. */
   Setscreen ( scrn, scrn, -1 );

   /* Restore original color palette. */
   Setpallete ( desk_palette );

   /* Return the reserved memory back to the system. */
   Mfree ( pic[1] );
   Mfree ( pic[0] );

   /* Close virtual workstation. */
   v_clsvwk ();
}
```

REVIEW

Smart Watch

Michigan Software
43345 Grand River
Novi, Michigan 48050
(313) 348-4477
$59.95 (retail)

Michigan Software's installation instructions are attached for your reference.

by E.H. Wysocki

The **Smart Watch**, a real-time clock module manufactured and distributed by Michigan Software, is a useful addition to the Atari 520ST. I recently purchased and installed this enhancement, and though the module works as advertised, I would give the following caveats to anyone considering its purchase and installation: The instructions packaged with the module are grossly inadequate to properly install the module (unless the buyer regularly works on Atari STs), and the module cannot be installed into a machine containing a RAM update.

The Smart Watch module is built in a modified 28-pin IC socket. The socket contains a circuit board and an IC chip. The bottom of the socket is potted with a black compound to seal the unit. The module comes with a 3.5-inch disk containing the necessary software, and three sheets of paper claiming to contain installation instructions.

Of the three sheets of paper, the first contains the product pitch and a 90-day limited warranty. The second sheet contains oversimplified installation instructions and a board diagram. The third describes software installation.

The installation instructions can be summed up as "crude." Unless the person installing the unit has had his ST apart before, it is likely that he will either be unsuccessful in his attempt to install the unit, or he will damage his machine by forcing it apart. To quote the instructions, "You will need a Phillips screwdriver to gain access to your main Mother Board of the 520ST; there are six screws that hold down the cover and additional screws that hold down the metal shielding plate. This plate must be removed as well."

The instructions neither mention removing the keyboard nor provide any information as to its proper removal. There is also no mention of the metal "twist-tabs" that retain the RF shield, let alone the fact that on early 520STs, the shield is soldered closed in two places.

I found the software installation simple and straightforward, except for the part about putting the clock reading program into an "Auto Folder." The instructions do not mention how one creates a folder and names it "AUTO." Your machine will recognize it as a run-first file and will automatically boot it on start-up.

A consideration that neither the advertising nor the literature packaged with the Smart Watch module mentions is the fact that if you intend to increase (or have already increased) your machine's RAM by using one of the commercial RAM updates (i.e., EASY ST RAM, etc.), you will have to choose between a clock or RAM. Because of space limitations under the metal shield, both modules will not fit. If this is the case, my suggestion would be to invest in a clock that would fit into the cartridge slot. Or, if you would prefer not to take your ST apart and peer into its brains, the cartridge slot type of clock would be an excellent alternative.

I am unable to offer an opinion as to the type of support that Michigan Software provides. On the third page of their instructions they state, "Should you still run into problems you may receive assistance by calling us at 313-348-4477." I attempted to contact their office at least five times in one week, at various times of the morning and afternoon. No one answered the telephone.

In conclusion, the Smart Watch does what the literature and advertising says it will do—it eliminates the need to set the date and time each time you start up your machine. However, the installation instructions provided with the module are inadequate unless the purchaser is experienced in taking the machine apart to reach its Mother Board. In addition, anyone considering this enhancement should also consider whether the clock module is more important than increased memory through a RAM update.

Edward Wysocki *is an engineer with over ten years of experience in machinery design and construction. He has worked primarily within the plastics industry, currently working at a Grand Rapids, Michigan, firm that manufactures plastic medical devices. He owns a 520ST for personal use. Though he has added memory enhancements, a ROM update, and built custom cables for the 520, his experience with computers has been primarily within an industrial setting (i.e., process controllers, machine-computer interfaces).*

REVIEW

ST Sound Digitizer

Navarone Industries
1043 Stierlin Road
Suite 201
Mountain View, CA 94040
(415) 624-6545

Medium or High Resolution
$139.95

by Andy Eddy

With the memory capacity, speed and power of the ST, many tasks that were previously delegated strictly to large-scale computer systems are crossing over to this "side of the tracks." Software for CAD/CAM/CAE (Computer Aided Design/Modeling/Engineering), high-quality graphics and animation, and digitizing (both video and audio) are appearing in great numbers for Atari users.

Digitized sound samples—such as those that appear at key moments in Firebird Software's **Starglider**, as well as many new entertainment releases for the ST—involve reading the analog input signal and converting the voltage level to numbers for storage, enhancement and playback by the computer. This is the same technology behind compact disks , though CDs are a much better medium for large storage of data. That's the problem with digitizing: For a good quality reproduction, you have to have a lot of memory for your sample. The more frequently your sound is sampled, the better the replication.

Navarone has licensed the **Hippo Sound Digitizer** from the now-defunct Hippopotamus Software, and they've got a pretty good product on their hands. It allows a sampling frequency between 1 KHz and 64 KHz (between 1,000 and 64,000 samples per second), the high number being the highest quality "recording." Of course, as we said earlier, on a 1040ST (with 1 megabyte of RAM) you are limited to an approximately 12-second sample at 64 KHz, but that figure will increase with a lowered sample rate; size is a trade-off for quality.

The cartridge that plugs into the ST cartridge port is exactly the same as Navarone's video digitizer and clock cart in size; they use the same casing for all their hardware add-ons to trim production costs. To suit its electronic requirements it contains two 1/8-inch phone jacks for Line In and Line Out, as well as two thumb wheel potentiometers for adjusting those levels. These are best set-up by watching the waveform on the real-time **O-Scope** (Oscilloscope) display, selected from the menu bar at the screen bottom.

On top of that, you'll have to equip yourself with a playback source (like an amplifier or stereo system), a microphone (if you want to record sounds or voices) and cables (one cable from your input source and one for your output to your playback device), because the package doesn't include them.

Conveniently, the manual briefly describes various microphones, adapters and cables that are available, along with their Radio Shack part numbers.

The **ST Sound Digitizer** software—which runs in either medium or high resolution—has two screens other than the above-mentioned O-Scope display: the Rack screen (which we'll get to later) and the Command screen, which is where the majority of the work is accomplished. Here you set the size of the sample (in seconds), the rate of sampling (in KHz), initiate the digitizing process and manipulate the end result in a variety of ways. They've even set you up with a folder full of sounds, like drum types and basic waveform types.

To find out where you stand at any given moment, you get a bar chart that displays the amount of available RAM broken down in three ways: space taken up by the sample, the space used by the copy buffer and the RAM left to use. Hitting the Stats button gives you a readout of the vital figures: the length of the sample (or, as you'll see later, a marked section) in seconds and number of samples (directly translated to RAM used), the sample rate and what percentage your high and low peaks are.

At the grass roots you can adjust the volume and overall level of the sample. You can mark a block and cut, copy, insert and replace it within the main body of the sound, in addition to saving and loading blocks of entire sound to and from disk. They've even slipped in a couple of commands for zooming in or out on the marked section of the sample to make your edits more precise. Overall, this cut-and-paste capability allows you to edit sounds together for whatever purpose you have, be it commercial applications or just for fun, giving you a low-cost recording and editing studio.

You can also alter the sound itself with the Reverse, Squeeze, and Stretch functions. The Reverse command is easy enough to understand: It will take the entire sound (or a marked block) and re-display it backwards. All those folks complaining of "backward masking" (those inverted messages inserted in musical passages that some claim are really evil, subconscious suggestions) will have an easier time of proving their claims with this tool.

Selecting Squeeze and Stretch short-

ens or lengthens a sound and changes its pitch equal to one musical note each time you enact them. If you double-click on either of these buttons, you can enter directly the number of times you want the effect to be processed. Unfortunately, there's no provision for changing the length of the sound without altering the pitch, so voice samples that are stretched or squeezed will tend to sound unnatural with regards to pitch, like the effect that spinning a turntable faster or slower will have on its playback.

Some hi-tech solutions have been devised for "compressing" a sound—in other words, changing the time of a sound without affecting the pitch—and some commercial producers, among others, are using it to cram more information into a limited period. I'm not sure what effect adding this procedure would have had on the cost of Navarone's unit, but it would have made the package that much more powerful by letting the user change the length of a sample without an audible difference.

There are a few more sound-processing tools available to you. The first involves a Mix command which will take a block and overlay it onto another area of the sample. Like "overdubbing" in a recording studio, this procedure combines the two signals into one. It'll make one voice become many voices or make various digitized sounds appear simultaneously created.

The other enhancements are on the Rack screen: namely, Echo and Reverb. Echo is a repeating of the initial sound (like yelling in a valley and hearing the sound come back) and reverb is a very fast echo (similar to singing in a tiled room and hearing the sound reverberate rapidly in the small area). You can add these effects to an existing sound in memory or use it to enhance a sound in memory or use it to enhance a sound as it is digitized.

The rates of the effects can be controlled as well. I found that these two enchancements, as well as the Mix command, severely alter the original sound's clarity, bringing about a fuzzier quality not heard at the start. For this reason, you should save the original sound, as you may find the result from these effects to be detrimental enough to be undesirable for your application.

Aside from digitizing a sound from an audio source, you can create your own by drawing the waveform and envelope on the screen. I'm not sure what use this is as it's ten times easier to record a sound with a mike or other source. If you choose to do things this way, you only have to click on the Draw Wave/Draw Env button on the Command screen and use the onscreen cursor to create the desired waveform and enveloping.

Another application that the ST Sound Digitizer comes equipped with—though, as you'll see, it's somewhat limited and I can't really see where you'd put it to use—allows you to plug a MIDI keyboard into the ST's MIDI ports for playback of sounds. I thought it strange that the manual would have you hooking a MIDI cable from the ST's MIDI IN port to the keyboard's MIDI OUT port because I figured that the digitizer software would send the sound to the keyboard. Reading further informs you that the software "does not provide any kind of sequencing or downloading function for your keyboard."

That's the problem with digitizing: For a good quality reproduction, you have to have a lot of memory for your sample.

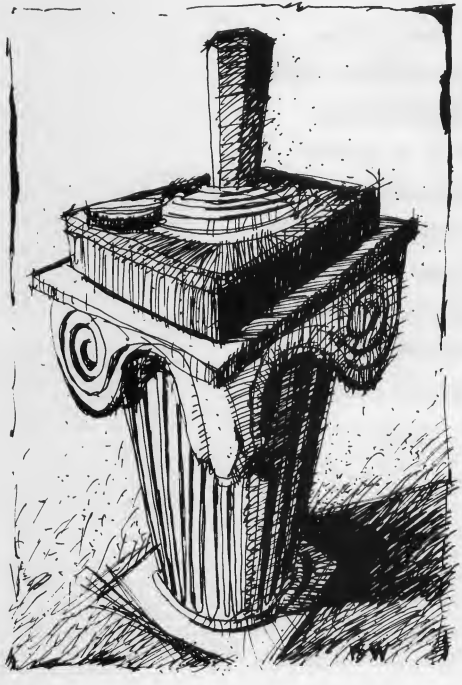
This is where the limitation comes in. Your MIDI keyboard will only *trigger* the ST, which holds the sample in its memory. I was hoping that it would be playable through the keyboard's electronics, like you have with a MIDI "patch" program (a utility that lets you move synthesizer sounds from the computer to a MIDI instrument and vice versa, as well as store them on a computer disk). The sound varies in pitch with the note you play (middle C is the key that plays the sound at its normal rate and is the keyboard's "pivot point"), so you can "play" on the ST, but the output is only monophonic (playing one note simultaneously), regardless of the capability of the keyboard. I doubt that this option would be usable in a performance. And again, you have to provide the cable.

Outside of the actual digitizer and software, the manual was found to be lacking in many respects. Granted, this package doesn't require much in the way of instruction; it's reasonably simple to use. What irked me is that some of the diagrams don't match their on-screen counterparts. An example of this is the Rack screen and its corresponding manual illustration on page 10 of the booklet. Similarly, on page 11, the Command screen buttons aren't labeled in the manual, and the overall layout varies a bit from the actual graphic. On top of that, much is left to the user's discovery, as the booklet is a bit thin on some points.

On the other hand, they've gone to quite a lot of trouble to include a good deal of technical data for those who'd like to utilize the digitizer in their programming efforts or those who'd like to know how it ticks. They've detailed how it works, the file format, even how to read the various voltages that are output by the cartridge.

The ST Sound Digitizer is none-too-high-priced and a neat piece of work on the whole. Complex works will require some other means as the length and quality of the sample are very restricted by memory usage, and mixing of multiple sounds is not the cleanest it could be. But if you'd like to build a library of sound effects for use in programs, or record and edit advertisements for your local radio station, this product fits the bill.

The Joy of Joysticks



Most of the would-be joystick magnates have long since moved to the 'Where Are They Now?' file, but a few hardy survivors remain.

by Bill Kunkel, Joyce Worley and Arnie Katz

Build a better mousetrap and the world will beat a path to your door. Since the debut of the classic Atari joystick (single-button, nine-position) almost a decade ago, a similar ideal has motivated artisans and technicians to better that original product.

The tinkerers tried everything: new technology; replacing the leaf-style switches with microswitches; new designs; moving the button; adding buttons; making the base larger; reducing the base; bigger sticks; smaller sticks; no sticks at all; and a near-infinity of similar modifications. During the glory days of the programmable video-game revolution, dozens of companies produced Atari-compatible joysticks of

every imaginable stripe. There were surfboard controllers; wireless controllers; no-hands controllers; tracballs (the upside-down mouse); and even podium-mounted joysticks.

Most of the would-be joystick magnates have long since moved to the "Where Are They Now?" file, but a few hardy survivors remain. Herewith is a look at some of the many controllers that are compatible with the ST.

Atari

It's interesting that the original Atari VCS joystick, the industry standard and one of the most universally successful products ever designed, has two distinctive features: It's square and it has a single action button. In all these years, the square shape and single button have worked just fine. True, some games were designed with two buttons in mind—Electronic Art's One-on-One, for example—but translating that or any other game to one-button format was never an insurmountable limitation. The box-like shape fit comfortably into the hands of a generation of game players, young and old alike.

When Atari finally got around to "improving" on this beautiful joystick, they gave it a trimline shape, added a button and called it the **Atari Pro-Line**.

The problem with dual buttons is obvious: They *must* be placed on the sides of the controller to make them accessible. As a result, the controller had to be thinned down so the user could grip the stick and get to both buttons.

This new stick is an insult to the pristine perfection of the old VCS joystick. After five minutes of play with this baby, the gamer's anchor hand is stiff, throbbing and fixed in an arthritic, claw-like shape. Using this stick is no walk in the park.

To round out the package, the designers also added an unwieldy nob to the top of the joystick shaft to make sure that the pain is spread evenly among hands *and* fingers.

Comfort—D; Control—C+; Durability—B+.

Kraft System

The **Kraft Atari-compatible Joystick** is the most perfect joystick ever made. It is the rich man's version of the classic Atari joystick. The cheap plastic casing of the Atari is streamlined into a one-piece (but still square) base; the fat, stubby shaft of the Atari is replaced here by a thin, gracefully beveled control stick; and the somewhat stiff play on the Atari is processed into the slickest control device in all the known worlds.

The **Kraft Joystick** is ideal for all kinds of games and users. It's amazingly durable too. This reviewer's own **Kraft Joystick** is over six years old, has been kicked, thrown and stepped on more than once, yet *never* missed a command in all that time.

Left-handers can opt for the "Switch Hitter" model, with duplicate action buttons in the upper left and right corners. For real playability, the newer versions contain a "mazemaster" feature. This disables the diagonal direction commands and limits movement to up-down/left-right for maze-chase contests.

Beautifully designed and crafted, this is the state-of-the-art controller, and has been for over half a decade.

Comfort—A; Control—A; Durability—A+.

Epyx

The **Epyx 500XJ Joystick** is a well made but badly designed joystick for right-handed players only (or lefties capable of using their left hand as anchor). The **500XJ** is built like a rock, with a thick, deep base and a microswitch-driven, solid-steel, nobbed shaft.

The action button is located on the right side, just above a cutaway. It is the position of this cutaway that makes this stick awkward for southpaws. It seems all but indestructible, and believe me, this puppy has been slammed up against the wall *more* than enough times to adequately test that supposition.

The problem is quite simple: This is *the most painful* controller ever built. The **500XJ** even hurts players with large hands or pianist's fingers. After a round or two of, say Robotron, you're ready for a trip to the Mayo Clinic.

Comfort—F; Control—B+; Durability—A+.

Mastertronic

The **Magnum Joystick** from the low-cost software maven, Mastertronic, is equivalent to most of the company's other products: It's a good, functional joystick at a reasonable price.

This controller uses a pistol-type grip, but mounts the button on the outside, rather than trigger-style. The short, nobbed shaft positioned on top of this housing makes the stick ideal for left- or right-hand use.

The **Magnum** is comfortable, and gives the user excellent control in all types of games. It is *not* all that durable, however. The red plastic control shaft pops out under significant torque. The shaft can be reinserted, but once it has popped free, the stick never functions exactly the same again.

Comfort—B+; Control—A; Durability—D−.

Suncom

Fans of "tight" joysticks — those with a minimum of play—invariably praise the Suncom joysticks. The **StarFighter** is a small joystick with a short, stump-like shaft, while the **Slik Stik** offers a nobbed control stick. Both have a very short "throw" (the physical distance the shaft must be moved to inaugurate a command).

The top-of-the-line Suncom stick is the **Tac2**. It features a metal (rather than plastic), nobbed shaft, Suncom's typically compact base, and dual-action buttons for southpaws (other Suncom sticks are right-handed).

Comfort—B; Control—B+; Durability—A.

Camerica

The Camerica joysticks are the ones you find in the bargain bin selling for as little as $7. They echo the worst of the joystickmania of the early '80s, with remnants of the poorest designs of the era, borrowed in style from the Wicos, PointMasters and GripSticks.

They have "bad" names like **Terminator, Turbo Charge** and **MicroMaster**, and all feature large, unwieldy bases and suction-cup "feet." These suction feet seemed like a good idea when first produced seven years ago, but the designers forgot one thing: Very few players have access to suction-compatible surfaces when playing video and computer games. Video games are often played on rugs, and computer games at workstations where the microprocessor generally occupies the space the joystick must be stuck to.

These joysticks also possess gigantic shafts, with finger-molded grips and optional fire buttons everywhere but the underside of the base itself. The **MicroMaster** and **Turbo Charge** have buttons atop the shaft, on the shaft itself and on the upper left and right-hand corners of the base, and *none* of them are comfortable.

As a wise man once said, "You get what you pay for."

Comfort—D–; Control—C; Durability—C.

ENTERTAINMENT

THE GUILD OF THIEVES by Magnetic Scrolls

Rainbird
P.O. Box 49
Ramsey, NJ 07446
(201) 444-5700
Low resolution $44.95

by Bill Kunkel

The Guild of Thieves is the eagerly awaited followup to Magnetic Scroll's original ST adventure, the Pawn. The Pawn broke new ground in text and illustrated-text adventures with its state-of-the-art parser and beautifully rendered pull-down illustrations.

Guild of Thieves, though not a sequel, reprises the parser (so sophisticated it can comprehend one word used many times in a single sentence, as in: "Plant the plant in the planter."); the setting (Kerovnia) and those breathtaking color "plates."

Unfortunately, the game's pilot is even flimsier than its predecessor's. The player, it seems, is up for membership in the notorious Kerovnian Guild of Thieves. The Guild doesn't accept just any old Tom, Dick or Thrushwhacker into its larcenous bosom, so the player must prove himself in matters of skill, stealth and treachery.

As the game opens, the player is sitting in a rowboat on a quiet, mist-speckled lake alongside a taskmaster from the Guild. The taskmaster supplies the player with a series of challenges—looting a mansion, despoiling and robbing a grave, etc.—which must be met to be accepted into the Guild.

Guild of Thieves uses the kind of nonstory line common to role-playing adventures. But unlike such games, the user does not see his surrogate develop and enhance characteristics and skills (strength, experience, spells, etc.) In other words, for an adventure, it's unusually thin. The player goes somewhere, steals something and returns with it. The taskmaster then tells him to steal something else, and so on until the requisite number of items have been pilfered.

On the plus side, the puzzles are often delightful and the illustrations are absolutely eye-popping. The mansion is particularly impressive, with its old-fashioned drawbridge, moat and billiard room. Each detail is rendered with incredible delicacy, from the soft brush-velvet of the overhead lamps to the burnished leather of the elegant couch.

One can only hope that in future fames Magnetic Scrolls will lavish half as much care on the story line as it does on the artwork.

LEISURE SUIT LARRY IN THE LAND OF THE LOUNGE LIZARDS by Al Lowe and Mark Crowe

Sierra On-Line, Inc.
P.O. Box 485
Coarsegold, CA 93614
(209) 683-6858
Low resolution $39.95

by Arnie Katz

Once upon a time, there was a harmlessly juvenile adventure game called **Softporn Adventure** (Sierra). This all-text title cast the player as a Don Juan on the town for a night of drinking, gambling and woman-chasing. Its somewhat unsophisticated view of sex, culled from the philosophy of Hugh Hefner, circa 1962, may have raised a few prudish hackles, but it was hard not to surrender to its lighthearted hedonism, liberally laced with broad, self-deprecating humor.

Leisure Suit Larry updates and expands upon the same theme. It's still not a terrific gift for the leader of the local chapter of NOW or a member of the Moral Majority, but it guarantees hours of lighthearted adventuring for more open-minded adults.

Designers Al Lowe and Mark Crowe present **Leisure Suit Larry** in the game system made famous by Sierra's **King's Quest** series of fantasy quests. The user moves 40-year-old Larry through more than 50 colorful, well-animated screens showing the pleasure and pitfalls of the sin city of "Lost Wages." Dozens of clever visual embellishments—like a barfly who swings her leg enticingly— constantly pop up to surprise and delight the player.

A 900-word parser helps the player communicate with the people (mostly women) encountered by the virginal salesman during his quest to win and woo an obliging lady.

Gambling also plays a big role in **Leisure Suit Larry**. A trip to the casino gives the user a chance to risk Larry's stake with a pull of the slot-machine lever or a hand of Blackjack. The ability to use the proceeds from such risk-taking to further Larry's pursuit of carnal knowledge makes the user a little more than careful about winning and losing imaginary money.

Leisure Suit Larry is one of a spate of recent adult adventure programs released for home computers. It's by far the best.

REVIEW

Someday, you'll be writing programs in several languages. You might as well plan for that eventuality now.

Metacomco Overview

The Programmer's Source:
a survey of development tools from Metacomco

Menu v1.2 $29.95
Metacomco Make v2.16 $69.95
Lattic C v3.04.01 $149.95
MCC Pascal 68000 v1.35.04 $99.95
Metacomco BCPL v1.11 $149.95
Cambridge Lisp v1.10 $199.95

Metacomco plc
26 Portland Square
Bristol BS2 8RZ, UK
(0272) 428781

by Tom Castle

So, you're interested in programming on the Atari ST. If you're not fluent already in a given programming language, you'll probably be asking which language is the best to learn. The answer is an emphatic "Any of them!" Each language has strong and weak points. Some are particularly useful for a specific application. Others are more generally useful with the concomitant sacrifices that go along with all-purpose products.

The general-wisdom advice floating around is to start with a language for which you can find help. BASIC has always been a popular choice, but C is quite in vogue. Pascal and its child, Modula-2, were expressly written as languages with which to learn programming. There are several, readily available sources of information and help, such as magazines, books, user

groups and bulletin boards for all those languages.

Once you decide on a language or if you already feel comfortable with a particular language, you then have to decide on which package you'll drop your hard-earned money. Again, for a little sage advice, look at the features, at how much support you'll get from the producer and the development community, and at price. You might also look at the future.

If you take up program development with a fervor, someday, you'll be writing programs in several languages. You might as well plan for that eventuality now. Metacomco, a major producer of the development tools for Motorola 68000-based machines, has anticipated your need. The development tools currently available from Metacomco for the Atari ST are **Menu+**, a GEM shell; **Make**, a UNIX-like utility for riding herd on projects; and the language products **Lattice C, MCC Pascal 68000, Cambridge Lisp, BCPL,** and the **MCC Macro Assembler**.

Not only do these products integrate well with each other, but several can be used to cross-develop on other computers. Metacomco's Lattice C is directly transportable to the IBM and Amiga versions of Lattice C. Metacomco specifically includes functions to provide UNIX, XENIX, and ANSI compatibility. Cross-development for the Amiga computer is particularly easy since Metacomco also produces MCC Pascal, BCPL, and Cambridge Lisp for the Amiga.

The Subject

For those of you new to compiled languages, a quick review of some of the terms should be in order. Each microprocessor, the Motorola MC68000 in this case, has a unique set of special memory spaces called registers. To manipulate data within those registers and within the main memory of the computer, operations must be programmed using a distinct instruction set. That instruction set is a set of numbers that the microprocessor will recognize as commands for it to perform given tasks. Programs written in this purely numerical code are called machine language. The command instructions are called opcodes. The data to be operated on are called operands.

An easier way of specifying opcodes and operands is called assembly language. For readability, the numerical opcodes are replaced by a set of mnemonics that each represent a given operation. Assembly language has a means of labelling constants, variables and points of a program. In addition, the other conventions are included to aid writing assembly language programs. Assembly language is considered low-level because it performs very closely to the actual workings of the computer. The conversions of an assembly language file into machine language is done by an assemblyer. The resulting machine language file is called an object or binary file.

The conversion of a high-level language into an object file is done by a compiler. Languages are termed high-level if they have the computer performing complex tasks in such a manner that the programmer doesn't have to worry about the details of how the task is accomplished. The text files that you write for a program, whether for an assembler or a compiler, are called source files. Usually a compiler will produce an assembly language or a pseudocode file before the final object file is produced. Each time a compiler must go over the text file to produce the object file is called a pass. So a single-pass compiler is easier, though presumably less versatile, than a multi-pass compiler.

Once a final object module is created, it must be treated by a linker program to make an executable program file. The linker usually connects some initialization code and any library routines that weren't resolved in all the object modules you specify.

Common Ground

If you do buy several development tools in the future, you may find yourself in the middle of a hodge-podge unless care is taken. Usually you can find an assembler that will produce modules that can be linked with those from high-level language compilers, but you may run into trouble linking modules from different high-level languages.

One of the nicest features of the Metacomco products is that any module produced by one product can be linked with modules from any other product. You have the possibility of writing modules in the language that is most suited to a particular section of your program and still come up with a unified package. If that isn't enough, all Metacomco products have the option to produce CP/M-68K object files for linking compatibility with Digital Research's LINK68 from the Atari Developer's System. Metacomco doesn't provide that linker with any of its products however. A nice feature of Metacomco's language packages is that they all include the assembler source files for the GEM functions.

Menu+

Each of the language products from Metacomco comes with a GEM shell called Menu+. You're able to customize drop-down menus that will perform the instructions you place in a batch file for each menu selection. This is one of the most useful products I have seen in a long time. If you have done much programming, you know that it's no fun hunting for your BATCH.TTP file among the myriad of files you build up in your compiler or linker folders.

Your customized menus and the associated batch files are coordinated through a file called MENU.INF. Prototypes of this control file come preconfigured for use with Metacomco's Lattice C, MCC Pascal, and MCC Macro Assembler. You can still alter these as you want.

Menu+ not only organizes your work space but also provides some handy utilities. A command line interface is provided for commands that aren't included with the drop-down menu. A history is generated every time you access Menu+. This isn't only provided for recollection's sake but also as a convenient method of repeating any command line by double clicking the particular line in the history. A UTIL.TTP program is included to perform functions like renaming, deleting, copying, showing, and printing files. This relieves the need to return to the GEM Desktop for those operations.

Menu+ doesn't have to be used solely with Metacomco's development products. It can be used with any development package; in fact, with any set of application programs. Although it's bundled with all of Metacomco's development packages, it's also sold separately. If you have a need to harness your desktop a little better, Menu+ is well worth the investment. It is very versatile and fairly inexpensive.

Metacomco Make

The other non-language develoment tool provided by Metacomco is their Make program. It's sold separately and

bundled with the Lattice C package. For those of you who are familiar with the UNIX Make, you'll feel right at home. It comes with a nice GEM-based screen editor, a prototype control file (called a makefile), a clock adjustment utility, and a Touch utility to update a file's date stamp without altering the contents.

Briefly, Make allows you to set up a programming project such that file dependencies are outlined and automatically managed by Make. For example, you might have a main program module that requires a separately linked module to be included. If the separately linked module is changed in any way, Make will detect the change, and recompile and assemble both the new module and the main program portion automatically.

Makefiles are generated by the user for each programming project. A variety of instructions are available for maximum flexibility in setting up a makefile. Explicit rules define file dependencies by the actual names of the files along with the exact command sequence you want followed. Implicit rules only consider file types when determining which files should be treated. Macros and control directives are also supported.

For those of you who are familiar with UNIX Make, there are some differences to be mentioned. Make uses makefiles rather than the built-in implicit rules of UNIX Make. The directives .DEFAULT and .PREVIOUS as well as the macros '$?' and '$%' are not supported by Make. The '::' form of explicit rules is not supported also. Tab characters are not essential as in UNIX Make, and colons must be separated by a space since colons have a special meaning in file path notation of GEMDOS.

Screen Editor

The screen editor included with the Make package is also included with all the language products. It uses drop-down menus as well as keystroke commands. **ED**, as Metacomco calls their multiple-window screen editor, uses a fixed-size text buffer which defaults to about 64K but can be changed easily. Scrolling can be performed horizontally or vertically with a message and command-line area at the bottom of the screen.

There's quite a variety of commands available with ED. Twenty-one immediate commands are peformed by either a single keypress or a control-key combination keypress. Most of those commands are used for cursor control, inserting and deleting text, and such. There are also 38 extended commands that must be typed at the command line. Those commands are used mostly for things like block operations, formatting, and search and replace functions.

The editor is quite useable, although the mouse is not supported in the text field. There is no provision for displaying line numbers.

Lattice C

Atari ST programming is heavily sided toward the use of C. This is most probably due to the fact that a lot of GEM was written in C. The DRI documentation and software in the Atari Developer's Package supported only the use of C. Although Lattice C was not one of the first C packages to be available for the Atari ST, the compiler already has a prominent reputation. It's considered by many to be the foremost C compiler for the IBM PC line.

Lattice C comes with several amenities not included with Metacomco's other language products. As mentioned before, Lattice C comes bundled with Menu+ and is the only language product that comes with Metacomco's Make and **Debug**+ programs.

It's also the only package that comes with a resource construction kit, NRSC, which enables the rapid development of resource files containing menus, dialog boxes and the like. Header files can be created with the program for C, Pascal, Modula-2, Fortran 77 formats. Metacomco also claims that no special programming is needed to conform the resource file to either medium- or high-resolution monitors. It should automatically adjust to the monitor in use. I don't have a monochrome monitor, so I wasn't able to test this claim. NRSC is upwardly compatible for resource files created by the Atari and **Megamax C** packages. It does have a few extras, however, that could jeopardize compatibility in the other direction. It supports the use of two additional flags that can be set by the user for extensible types of objects.

A symbolic debugger, disassembler, and link loader called Debug+ is provided with the Lattice C package. You can display memory as hexadecimal and ASCII dumps or as a disassembly listing. The debugger uses the symbols generated at the end of a program by the -DEBUG linker option. The link loader will install the object module and any library modules into memory, resolve intermodule references in preparation for debugging. Debug+ has a rich set of commands that let you display the entire symbol table, dissassemblies, dumps, and memory locations of any symbol. User-selectable breakpoints can be inserted in your code, or you can use the STATE command to examine register contents and the current assembly instruction. Any variable can be examined at any point in the debugging process. Debug+ also lets you use predefined or user-defined macros to speed up the debugging process. A WHERE command will tell you what C function is currently being examined. Commands are also provided to trace forward and backward through a program. Thirty-one commands are provided altogether.

The Lattice C compiler is complete. The full Kernighan and Ritchie specification is included with extensions to conform to the ANSI standard. As mentioned earlier, functions providing both UNIX and XENIX compatibility are included. In fact, there are so many functions included in the runtime library, a little over 320 in all, you could easily be seduced into using alluring functions that will destroy any chance of compatibility with other C compilers for the Atari ST. This isn't good for your *ST-Log* submissions. In fact, compatibility with other C compilers for the ST must be meticulously handled since Lattice C is the only C compiler that uses 32-bit integers. Careful, careful!

Even with the caveats, it's a nice compiler. Line A calls—the code that GEM's VDI uses to draw graphics primitives, to manipulate the mouse form, to perform bit and text block transfers, and to manipulate sprites and rasters—are included. Double-precision floating point as well as MC 68881 Math Coprocessor support is available.

There are three levels of memory allocation which can be called. Level three functions call level two functions, and so on. There are also three levels of file access. GEMDOS, BIOS, and XBIOS system traps are supported. Qsorts for all the variable types are included. A full set of string manipulation functions are also included.

You can easily be overwhelmed by Lattice C if you're not already familiar with C. However, you'll never outgrow this product. The 600-page manual covers the compiler, linker, and all accessory programs in great detail. Each function is thoroughly described, many with example code fragments for the more complex functions.

MCC Pascal

Pascal was originally written by Niklaus Wirth in the late '60's based on the popular language ALGOL. Wirth formulated the language as a teaching tool for the instruction of proper, structured programming techniques. Large programs are segmented into structured blocks which can be nested, but not overlapped. The block structure encourages "top-down" program development. A skeleton can be written while the various levels of detail can be tackled as you come to them. Pascal has since become one of the most widely used languages. It's one of the most readable and readily accomplished languages. This is testified by the abundance of Pascal packages for the Atari ST.

MCC Pascal 68000 offers a single-pass compiler that produces native 68000 code rather than a p-code. As mentioned earlier, this allows Pascal object files to be linked with object modules from any of Metacomco's other languages. A rather complete error reporting system is included.

The implementation also conforms to the ISO 7185 (level 0) standard. This is important for portability considerations. Very large arrays and sets are limited only by available memory. MCC Pascal has just about as many classifications of variables, imaginable, including 1- and 2-byte integer subranges, variant records, booleans, 4-byte pointers, and file types among others.

Although all the GEM VDI and AES bindings are accessible, no resource construction set is included in the package. Like the Lattice C compiler, integers are expressed as 32-bit values. MCC Pascal also uses 32-bit, 7-digit precision real numbers. Indentifiers can be of any length, and all characters are considered.

There are a few extensions to the ISO standard included with MCC Pascal. The RESET and REWRITE procedures are included so that internal files, those that exist only during the execution of the program, can access named files. The INCLUDE and EXTERNAL directives allow modular inclusion of program fragments during the compilation process. Shades of Modula-2! MCC Pascal wasn't extended to include bitwise operators however.

The MCC Pascal manual disclaims the idea of being a language tutorial. It is, however, a nice explanation of the language elements and syntax. Three example programs are given which show how to access GEM features, to call assembler language routines from Pascal, and other nifty things.

MCC Macro Assembler

Now that you feel comfortable with the popular high-level languages, it's time to delve into the bowels of programming, the assembler. The MCC Macro Assembler package, like the Pascal product, disclaims the idea that it is a tutorial. This is an understatement. The manual does not even list the 68000 instruction set. It does, however, contain a fairly thorough treatment of addressing modes, coding conventions, and syntax. The assembly control directives, along with the facility for conditional assembly, are thoroughly explained. There is also a bouncing-ball example program and the source code for the GEM routines.

Several options of the assembler control file output. The MCC Macro Assembler can produce GST or TOS format object files or simply a listing file. Symbol dumps and cross-reference tables can be generated also. You have the option to make the assembler insensitive to label upper/lower case distinctions.

The source code of a simple debugger is included. This seems to be more for the purpose of illustrating some TOS features than for the purpose of providing a highly useful debugger.

There are some ups and downs to the MCC Macro Assembler. Local labels are supported. That's good. No macro libraries, other than the GEM VDI and AES stuff, are provided. That's bad. One nice thing is the facility to include files by the assembler directives IIDR (header), EQU (equate), and INC (directories from which INCLUDE files should be searched).

BCPL

Well, maybe you're not quite ready for assembly-language programming but want the enormous power that "low-level" access can bring to you. The C language provides access to the inner workings of the computer, but it's not the only alternative to assembly language programming. There's a predecessor to C that's even less restrictive to the programmer than C. It's called BCPL. I had never heard of it. My friends had never heard of it. It seems to be widely used though. Metacomco wrote AmigaDOS in BCPL. It must be like Slim Whitman—it's so darn popular in Europe.

If Pascal is the fascist of variable typing, BCPL is the anarchist. BCPL uses data words of 32 bits. It doesn't care how you want to use them. They can be pointers, integers, packed characters, booleans, or whatever bit pattern you want.

There is some structure to variable usage, however. From the vein of BCPL flowed the C tradition of strings as pointers to the byte-by-byte pack. Unlike C, the memory occupied by the character array starts with the length of the array followed by the characters. BCPL also incorporates vectors which are arrays of data other than character strings. Tables, which are initialized vectors, are also used.

Like C, BCPL passes parameters by value and controls them by scope. To be specific, variables can be Manifest, being a globally defined constant; Local, existing only in its prescribed block or procedure and being deallocated upon exit from the block or procedure; Static, retaining its value throughout the execution of a program but having a local scope; or Global, being available at all times.

Metacomco's BCPL manual is probably the most instructive of its language products as to the explanation of the language elements. The BCPL environment and memory-usage map are given in an appendix. Six example programs are given, exemplifying input/output, switch-case constructs, data manipulation, flow control, and GEM functions. A detailed description of each library function is provided in the manual.

Metacomco includes a few extensions to the language. Floating point operations are not usually a feature of BCPL, but Metacomco provides single precision real numbers and a few functions that allow real integer conversions. A WORD indirect operator is provided to manipulate arrays of 16-bit values like the control arrays returned from initializing a GEM workstation. The EXTERNAL Keyword is included with

BCPL, like Metacomco's other language products, to allow interfacing to C and Assembler routines. They have also added procedures for those of you who would understand the importance of that.

Cambridge Lisp

The last language product for us to examine is Metacomco's Cambridge Lisp. With all the hoopla over artificial intelligence over the last few years, you might think that Lisp is a fairly new language. The truth is that Lisp was invented by John McCarthy in the late '50's. We're talking vacuum tubes here. Lisp stands for LISt Processing. It relies heavily on recursion and the basic data structure, a pair. From those two ideas, lists and trees can be created and manipulated.

Lisp is used in a variety of applications that are not well-suited to procedural languages. Natural language query interfaces, symbolic algebra, robotics, and expert database systems are all fertile fields for Lisp. In fact, any application that's heavily decision-based is ripe for Lisp.

There has been sufficient time since the inception of the original Lisp for several dialects to emerge. Metacomco describes Cambridge Lisp as a member of the standard Lisp family with close similarities to PSL, Portable Standard Lisp. A nice feature of the manual is to point out differences between Cambridge Lisp and other Lisp systems, particularly MacLisp, Interlisp, and Common Lisp. An appendix lists the extensions available in Cambridge Lisp that are not found in standard Lisp.

There are two considerations that set Lisp apart from Metacomco's other language products. First, it's basically an interactive language. Most of your interaction with Lisp is through the Supervisor, a read-evaluate-print loop interpreter. Anyone who has programmed in BASIC will attest to the ease of program development under an interpreter rather than a compiler-based system. Cambridge Lisp offers both.

Second, it's a declarative language. Rather than setting down a procedure for the computer to follow, you give the computer the means to process basic data structures and evaluate expressions while letting the computer figure out how it will go about the task.

Cambridge Lisp is a large language. There are about 400 standard functions along with the facility to define your own functions. That does not include the GEM routines. You can easily see that Lisp was originally developed for mainframe computers. Metacomco suggests a full megabyte of RAM in your ST to take full advantage of Cambridge Lisp, but the system is still quite useable on a 520ST.

There are several facilities that enhance Lisp implementation. Cambridge Lisp provides full garbage collection and error reporting. A built-in prettyprinter is provided. Cambridge Lisp also provides the facilities to access double precision IEEE format-floating point numbers, rational numbers of the form (27/513), and trigonometric functions, and to use several input/output streams, and to customize syntax.

Cambridge Lisp also has some amenities to make programming chores easier. The old program feature of Lisp that allows a section of code to be enclosed with loops and local variables is retained. The catch and throw functions are also available to make non-local jumps. A system for creating includable modules is also present.

Error handling and debugging is made easier with some of Cambridge Lisp's functions. A few functions let you retain a moderate amount of control over the program even when serious mishaps occur. You can customize the amount of error reporting and how much backtrace information you want for given situations. Cambridge Lisp also provides several functions to perform trace, history, and logic mapping operations. Tracing can be performed in either interpreter or compiler mode.

I won't attempt to give a synopsis of the language itself. That would be a monumental task. Lisp is different from all the FORTRAN descendents with which you may be familiar. It's probably sufficient to say that learning Lisp will not be easy, especially if you are already in the procedural mindset of C, BASIC, Pascal, or assembler language. It would also be safe to say that once you are comfortable with Lisp, you'll be able to write efficient programs that can't be written easily in another language.

Conclusions

Metacomco has a fairly diversified line of programming-development tools. The Menu+ program is especially useful even if not used or purchased with one of the language packages. Make, as any UNIX programmer will tell you, is nice for those large projects. However, if you're a middling dabbler like me, it may not be worth the expense and time to use.

The products that Metacomco should release as stand-alone packages are the resource file editor (NRSC) and the debugger (Debug+), which are included with Lattice C. The unavailability of NRSC and Debug+ to the Pascal, BCPL, Lisp, or assembler language programmer makes those packages look less attractive.

Metacomco's strong suit is Lattice C. They've included everything you would need for a professional development system. This isn't surprising since Atari/DRI promulgated C as the language of choice for GEM and the Atari Developer's Kit.

Lattice C should do well since it offers IBM and Amiga compatibility. The **Alcyon C** compiler from the Atari Developer's Kit will provide IBM compatibility under GEM, but provides no support for XENIX or UNIX.

The Lattice C manual is somewhat bug-ridden. The first 30 pages of mine were scrambled. The manual also, on occasion, uses British spellings for GEM functions like form_centre, but the linker will spit these back out to you. Overall, though, it's a good package, and a strong contender in the C market.

The strong points of Metacomco's other languages are the common linker and the ability to compile code to either CP/M-68K or GST format object files. However, few people will currently need several languages that produce compatible object modules. Initially, an assembler that's compatible with your higher level language may be all that you need. It's comforting to know that it's available, though. In fact, who's to say that you won't need that sort of latitude some day? It gives you some peace of mind that it's possible to expand your language repertoire with the confidence of compatibility.

Tom Castle *is an M.S. Chemist in Kalamazoo, Michigan. He bought his first personal computer, an Apple II, in 1980, but had no hesitation switching over to his Atari 1040ST. He spends most of his programming efforts using C on the Atari and Turbo Prolog on IBM machines.*

REVIEW

Plundered Hearts

Infocom
125 Cambridge Park Drive
Cambridge, MA 02140
(617) 576-3190
$39.95
520ST—Low Resolution

by Betty D. DeMunn

At last! An Infocom "Interactive Fiction" text adventure of a woman by a woman! News of this incredible breakthrough came as a delightful surprise. For years I've been masquerading as a macho male 18 year old in countless adventures, so the chance to shed "him" and become "her" was long overdue and more than welcome. Ripping open the package in a frenzy of anticipation, I thought, "Bless you, little Amy Briggs. You have taken one small step for woman, one giant step for womankind."

As usual, I found goodies in the package: a velvet reticle (purse), a letter from Jean Lafond, and a banknote, plus a slim manual containing basic info. Games are saved within the story disk, so I was all ready. With trembling hands and pounding heart, I booted **Plundered Hearts**.

Set in the 17th century, the story is obviously a spoof of the romance nov-

els that leer from drugstore racks. You are a young gentlewoman on a ship in the Caribbean, sailing to your dying father's bedside as per instructions in the letter. En route you are boarded by pirates (Oops, Freudian chemise)—the *ship* is boarded by pirates, and you are snatched by a gorgeous hunk named Nicholas Jamison. Your pulse races and your bosom heaves. He sets you straight about Governor Lafond, a scurrilous liar and champion villain who actually holds dear Papa prisoner on the island of Ste. Sinistra. Your mission is to rescue Papa, decimate the evil Lafond, and escape with Nick. On the way to this happy ending, you encounter a lot of heavy-breathers, a crocodile, and several fates worse than death. Sounds like fun, huh?

Did you ever slip on a banana peel? You know, airborne one moment and flat on your keester the next? That's the feeling I experienced as this adventure unfolded. It was like playing hide-and-seek alone. I could find me every time. In fact, Plundered Hearts went by so fast, I didn't even catch the heroine's first name.

Hype on the package proclaims that Miss Briggs read hundreds of romance novels, researched 17th-century costumes and ships, and was wooed by a dashing pirate. No doubt this is true. But Amy contracted cliche-itis along the way, and I seemed to have all the antidotes. I finished ("solved" is too strong a word) Plundered Hearts in four hours, 523 moves. It should have been faster, but I kept trying to make a ballgown out of drapes, and putting hoops in my skirt so I could parachute out of the crow's nest! When one spends $40 for an adventure game, one expects to develop a meaningful relationship with it. This was a case of Wham-Bam, etc.

A new title by Infocom always meant buying a notebook to fill with maps, clues and assorted ravings. I've completed only four out of 20-odd adventures without the aid of Invisiclues. I'm self-rated as "Almost-Intermediate," and of average intelligence. If Infocom still printed difficulty ratings on the package, Plundered Hearts would have to be assigned: "Beginner — Novice — E.T."

Let it be noted, I love Infocom so much that my four-letter ATM code is "G-R-U-E." But I fear it has loaded its marketing bow with a horseshoe. Women who read romance novels would be better off buying ten of them for the price of Plundered Hearts. Not only that, but women who own, or have access to, or any interest in computers are likely to prefer other types of fiction.

Let's not be too cruel. Plundered Hearts is well-written. The brilliant purple prose is amusing and often funny. Amy Briggs is a talent, but, in my

You are a young gentlewoman on a ship in the Caribbean, sailing to your dying father's bedside as per instructions in the letter. En route you are boarded by pirates.

opinion, wasted on this sexist plot. Sexist because one of the levels you attain on your way to Happy Ending is "Lady Leman." Sexist because it's difficult to relate to a woman who doesn't know one end of a rapier from another and isn't allowed to handle a pistol or to sharpen her dagger. Granted, she rescues her virile lover from certain death three times, but doesn't that hint of his latent wimpiness? The ending suggests that the happy couple will sail to America for the sequel. I can wait.

Perhaps women are not as sadistic, manipulative and devious as men. If those qualities make for exciting, challenging adventure games, then harden up, Amy. Abandon the gentle, nurturing nature that makes us women and let the sequel be convoluted and frustrating. Give us months of pondering and a reason to buy the hint book.

A note to male players: You must be totally liberated to play Plundered Hearts, but go ahead, it'll do you good!

A few uneasy questions remain. Why was "About the Authors" omitted from the manual? We want to know more about Amy Briggs and her background.Why was Plundered Hearts kept secret until its release? *Status Line*, the Infocom newsletter, usually hypes upcoming titles, but I don't recall reading anything about Plundered Hearts there, or anywhere else, for that matter. Who is Amy Briggs, really?

Finally, I'd like to remind you that never within recent recorded history has there been a negative review of an Infocom game. Their track record has been a miracle of success. But there's a first time for everything, and Plundered Hearts disappointed me. The intent is to be applauded. Women have long been overlooked, both as authors and consumers, but to grab us, you need a stronger hook. Let us be what we are today, or will be in the future—not what we were 300 years ago. Or, if you must spoof, make the spoof challenging enough to flatter our intelligence. Even a first-time player would know what to do with a sliver of mirror, a library, a chandelier. Those tricks have been done and done and done in movies, TV, and books.

Being a feisty old feminist, I have to say that Plundered Hearts is one small step for womankind, sideways.

Betty D. DeMunn *is a professional actress and freelance writer who lives in Buffalo, New York. She's been addicted to Ataris since 1982, when a 400 followed her home one day, and grew up to be a 520ST. Other hobbies include: one husband, five children, seven grandchildren, and one great-grandson, Nick. Wow!*

A Text Enciphering Program

Part Two

ASSEMBLY LINE

by Douglas Weir

The phrase, 'I wandered lonely as a cloud' will be enciphered differently from 'I wander lonely as a cloud.'

You may still be wondering what those cryptic words "uoou mcempun" at the end of last month's installment meant. This month we'll add to our program the routines needed to decipher that message, as well as smooth a few rough spots in the original program.

Lost in spaces

If you typed in, assembled, and ran last month's program, you probably noticed that the same text is enciphered differently depending on the number of spaces that are typed between (or, for that matter, in the middle of) words. Tabs, which are handled similarly to spaces, had the same effect. This wouldn't necessarily be a flaw if there were some way to work variable-sized space-groups into our enciphering scheme—in that case, it would be a feature. But since I didn't do that, we'll have to call the feature a bug.

What would be the ideal way to handle spaces? Assuming we're going to use them as separators of words, we don't want to filter all spaces from input. On the other hand, one space is always enough as a separator for our purposes, so we don't need to reproduce groups of them. But even where we save single spaces from input (plain text) into output (enciphered text), the clean way to handle them would be not to process them at all—simply print them. At first glance, it seems that's exactly what was done in last month's program.

Keep in mind that our encipherment scheme involves cycling regularly through a table of numbers (**incs**) that are added to the plain-text converted ASCII codes. The phrase "I wandered lonely as a cloud" will be enciphered differently from "I wander lonely as a cloud"; after the "wander," the remaining characters will be affected by different numbers from the **incs** table, since the characters have changed, but the numbers (and the period of cycling) have not. In fact, that's what makes our ciphers somewhat more difficult to crack than the straight-substitution variety . . . we hope.

In last month's program, spaces were not actually enciphered; they were simply passed through and printed as is. But every time a space was read, the **incs** table cycle was advanced one step (even though the current increment wasn't used that time), and the effect was the same as if a valid character had been read and enciphered. That meant that the number of spaces, consecutive or not, made a difference in the enciphered text. The phrase

```
Once upon a midnight dreary
yxmj zrqp h vrmojhrd iwjcta
```

would be enciphered differently from

```
Once  upon  a  midnight  dreary
yxmj  wrqu  h  vreojqrd  iwgctf
```

(there are two spaces after "Once" in the second example; otherwise the two are identical).

What we want to do is make the enciphered output come out the same no matter how many spaces occur anywhere in the input. This turns out to be pretty easy to do. In last month's program, when spaces or tabs (which are the same thing as far as we're concerned) were detected, execution branched to the label **e__raw** and almost all further processing was skipped . . . almost all. The cycle counters (in registers **d3** and **d2**) were still advanced. All we have to do is move the **e__raw** and the line of code that follows it to the location just before the label **e__test**, and change the couple of branches to **e__test** to branches to **e__raw**. Now the cycle counters will only be affected when a valid character has already been processed. You can see these changes in this month's listing (the labels now begin with **x__** instead of **e__**).

The other thing we'd like to do, as mentioned above, is copy only the first of a consecutive group of spaces into the output, and ignore the rest. This is accomplished by setting up a "flag" variable. The flag is initialized with the value zero. Every time a valid character is read, *0* is written into the flag. As soon as a space is detected, the value *1* is written into the flag. From now on

whenever a space is detected, the flag is checked. If it has a value of *1*, then we know that the previous character also was a space, and this one can be ignored. If the flag has a value of *0*, then this must be the "first" space, and it should be echoed in the output.

The flag variable can be a location in memory, just like **last_ch** in last month's program. However, it takes time (and sometimes an extra instruction or two) to access memory locations, and using a register (if there's an ideal one available) is much more convenient. In this month's program, the register **d7** is used as the "space flag" in the subroutine **x_cipher**.

Finally, a couple of other minor changes result in tabs being handled in exactly the same way as are spaces.

What a different arrays make

As for deciphering the enciphered text, there's not much to it beyond what we've already done. In fact, you could use the original code in **en-cipher** from the last time, and instead of the last four lines of code under the label **e_nxt2**, type in these:

```
sub.w   d1,d0           subract increment
bcc.b   e_nxt3          if ≥0, continue
addi.w  #C_MAX,d0       else wrap around top
```

In other words, subtract the current increment from the enciphered text (thus reversing the enciphering process). Then you check to see if the result is a number less than zero. That wouldn't be good, since it wouldn't give us a valid array index (remember, we're going to index into the **ciphers** array). We saw last time that any time a larger number is subtracted from a smaller, the 68000's Carry flag is set. So if the Carry flag is cleared, everything's okay, and we go on to index a deciphered character. Otherwise we simply add the size of **ciphers** to the negative number we know is in **d0**. The result is the same as if we had indexed down to the "bottom" (element 0) of ciphers and "around" the "top" (element 25, the exact opposite of what we did when enciphering.

Making dual with one

We now have all the ingredients we need to both encipher and decipher text. Since so much of the code for the two operations will be identical, it makes sense to combine them into a single program and let the user choose which of the two he or she wants to do. And that's that I've done in this month's program.

The subroutine **encipher** has been replaced by **x_cipher** in the new version. Despite all the changes mentioned above, the two routines are still very similar. Two more data registers, **d6** and **d7**, are used. The first, **d6**, contains a value that tells **x_cipher** whether it's supposed to encipher or decipher its text. The second, **d7**, is used as the "space flag" discussed above. Notice that now all the data registers are saved at the beginning of the routine and restored at the end, even though **d5** isn't used for anything. If you think that it would make

'Keep hacking' came out as 'uoou mcempun' last time, but now it's enciphered as 'uoou mfemkun.'

sense to use **d5** (instead of the memory location **last_ch**) to hold the last character typed, you're right. Last month **last_ch** was there to introduce the use of variables, but a register would be faster, and it would save time and space, too. Unfortunately, writing it in wouldn't have helped me save time making the dealine for this issue of *ST-Log*.

A second subroutine, prompt, has been added to handle the task of finding out what the user wants to do—encipher or decipher text. The prompt message is printed, and then the GEMDOS **conin** function is used to read the keyboard for an **e** (encipher) or a **d** (decipher text)—uppercase letters are converted to lowercase by the lines of code immediately preceding **p_nxt0**.

As soon as a valid response is received (otherwise **prompt** simply keeps repeating **prompt_msg**), one of two values (**ENC** for encipher, **XDC** for decipher) is written into **c_flag**, and another carriage return and line feed are written to the screen, so that the user will have a clean screen line to type on. This "newline" string is simply the end of **prompt_msg**, with its own label, **prompt_end**. The GEMDOS routines recognize only a null (binary zero) as a valid string terminator, so you can label "interior" parts of a string to your heart's content. We did something similar last month with **p_string** and **c_string**. When **c_string** is passed to GEMDOS function 9, only its contents are printed (remember, we added the null to the end of **c_string** at the very end of encipher). When **p_string** is passed, a carriage return and line feed are printed (thus moving the cursor down to a new screen line), and, since there's no null (i.e., **0**) until the "end" of **c_string**, GEMDOS continues merrily along and prints all its contents too.

With the space and tab-handling improvements discussed above, you'll find that this month's program enciphers text a bit differently from last month's version. For example, "keep hacking" came out as "uoou mcempun" last time, but now it's enciphered as "uoou mfemkun," no matter how many tabs and spaces you type between or within the words.

There is still a lot that can be done with this program. One obvious and simple enhancement would prompt the user at the end and allow him or her to re-run the whole thing over again, as many times as desired. The amount of text processed could be increased, and screen editing could be added—I hate not being able to backspace over a mistake and retype it. File storage and retrieval could be added. A protocol for having an enciphered message contain its own "private" set of increments in a header section could be developed. All characters—not just letters of the alphabet—could be processed. Or some effort could be put into simply making the existing program more efficient, compact, and elegant.

```
*************************************************************************
*
* A Text Enciphering Program-- Part 2
*
*   by Douglas Weir
*
*   Copyright 1988, ST-Log
*
*************************************************************************




        text

        bsr             prompt                  determine mode

        move.l          #c_string,-(a7)         string space address
        bsr             x_cipher                encipher a string
        addq.l          #4,a7                   pop arg

        move.l          #p_string,-(a7)         print string address
        move.w          #9,-(a7)                code=print string
        trap            #1                      do it
        move.w          #0,-(a7)                code=exit program
        trap            #1                      do it

*************************************************************************
*
* x_cipher-- encipher or decipher a text string.
*
*    at entry:
*             (a7) + 4 => space for storing processed string.
*
*    registers used:
*             d0 -- character from keyboard
*             d1 -- element from "incs"
*             d2 -- index into "incs"
*             d3 -- increment phase counter
*             d4 -- main loop counter
*             d5 -- not used
*             d6 -- encipher/decipher flag
*             d7 -- space flag
*
*             a0,a1 -- not used (affected by GEMDOS calls)
*             a2 -- not used
*             a3 -- base pointer to "ciphers"
*             a4 -- pointer to string space
*             a5 -- base pointer to "incs"
*             a6 -- frame pointer
*
*    at exit:
*             all registers preserved.
*
*************************************************************************

x_cipher:
        link            a6,#0                   frame pointer
        movem.l         d0-d7/a0-a5,-(a7)       save registers

        movea.l         8(a6),a4                point to string space
        movea.l         #ciphers,a3             point to cipher table
        movea.l         #incs,a5                point to increments
        clr.l           d2                      reset inc index
        clr.l           d7                      reset space flag
        move.b          c_flag,d6               get cipher/decipher flag
        move.w          #I_PHASE,d3             reset phase counter
        move.w          #S_SIZE,d4              absolute counter
        move.b          #CR,last_ch             initialize last char
x_loop:
```

```
        move.w      #1,-(a7)            code=conin
        trap        #1                  do it
        addq.l      #2,a7               pop arg
        cmpi.b      #CR,d0              carriage return?
        beq         x_end               if so

        cmpi.b      #SPACE,d0           space?
        beq.b       x_space             if so
        cmpi.b      #TAB,d0             else: tab?
        bne.b       x_nxt00             if not
x_space:
        cmpi.b      #1,d7               more than one?
        beq         x_test              if so, skip everything
        move.b      #SPACE,last_ch      else save it
        move.b      #1,d7               and set flag
        bra         x_raw               and write it
x_nxt00:
        move.b      #0,d7               reset space flag
        cmpi.b      #BS,d0              backspace?
        bne.b       x_nxt0              if not
        move.b      last_ch,d0          else get last char
        move.w      d0,-(a7)            push it
        move.w      #2,-(a7)            code=conout
        trap        #1                  do it
        addq.l      #4,a7               pop args
        bra.b       x_loop              and start over

x_nxt0:
        move.b      d0,last_ch          save raw char
        cmpi.b      #A,d0               lower than 'A'?
        bcs.b       x_raw               if so, just write it
        cmpi.b      #Z1,d0              else: upper case?
        bcc.b       x_nxt1              if not
        addi.w      #U_CONV,d0          else convert to lower case
        bra.b       x_nxt2              and continue
x_nxt1:
        cmpi.w      #a,d0               non-alphabetic?
        bcs.b       x_raw               if so, write it
        cmpi.w      #z1,d0              else: lower case?
        bcc.b       x_raw               if not, write it as is

x_nxt2:
        subi.w      #C_CONV,d0          else make it an index
        move.b      0(a5,d2.w),d1       get current increment

        cmpi.b      #XDC,d6             decipher?
        beq.b       x_nxt3              if so
        add.w       d1,d0               else encipher by adding inc
        cmpi.w      #C_MAX,d0           over maximum?
        bcs.b       x_nxt4              if not
        subi.w      #C_MAX,d0           else correct it
        bra.b       x_nxt4              and continue
x_nxt3:
        sub.w       d1,d0               decipher by subtracting inc
        bcc.b       x_nxt4              if >= 0, continue
        addi.w      #C_MAX,d0           else wrap around top

x_nxt4:
        move.b      0(a3,d0.w),d0       get enciphered code
        subq.w      #1,d3               decrement phase counter
        bne.b       x_raw               if phase not exhausted
        move.w      #I_PHASE,d3         else restart counter
        addq.l      #1,d2               and increment index
        cmpi.w      #I_COUNT,d2         over maximum?
        bcs.b       x_raw               if not
        subi.w      #I_COUNT,d2         else wrap around

x_raw:  move.b      d0,(a4)+            write it
x_test: dbra        d4,x_loop           go till end

x_end:
        move.b      #LF,(a4)+           append line feed
        move.b      #CR,(a4)+           and carriage return
        move.b      #0,(a4)             and null

        movem.l     (a7)+,d0-d7/a0-a5   restore registers
        unlk        a6                  deallocate frame
        rts                             and return
```

```
*****************************************************************
*
*    prompt-- determine function: encipher or decipher.
*
*    at exit:
*            registers d0-d1, a0-a1 are changed.
*
*****************************************************************

prompt:
        move.l          #prompt_msg,-(a7)       prompt message
        move.w          #9,-(a7)                code=print string
        trap            #1                      do it
        addq.l          #6,a7                   pop args

        move.w          #1,-(a7)                code=conin
        trap            #1                      do it
        addq.l          #2,a7                   pop arg

        cmpi.b          #a,d0                   is letter < 'a'?
        bcc.b           p_nxt0                  if not
        addi.w          #U_CONV,d0              else convert
p_nxt0:
        cmpi.b          #e,d0                   'e' = encipher?
        bne.b           try_dec                 if not
        move.b          #ENC,c_flag             else set flag
        bra.b           prompt_out              and leave
try_dec:
        cmpi.b          #d,d0                   'd' = decipher?
        bne.b           prompt                  if not, keep trying
        move.b          #XDC,c_flag             else set flag
prompt_out:
        move.l          #prompt_end,-(a7)       newline
        move.w          #9,-(a7)                code=print string
        trap            #1                      do it
        addq.l          #6,a7                   pop args
        rts                                     and return

*****
*
* data area
*
*****

        data

prompt_msg      dc.b    'Type <e> to encipher, <d> to decipher'
prompt_end      dc.b    10,13,0

last_ch         ds.b    1               holds last char typed
c_flag          ds.b    1               signals cipher or decipher

p_string        dc.b    10,13           line feed, carriage return
c_string        ds.b    100             space for enciphered string
S_SIZE          equ     *-c_string      length of string

ciphers         dc.b    'abcdefghijklmnopqrstuvwxyz'
C_MAX           equ     *-ciphers       length of table
C_CONV          equ     97              'a' - 97 = 0
U_CONV          equ     32              'A' + 32 = 'a'

incs            dc.b    10,5,2,7,9,1    ciphering increments
I_COUNT         equ     *-incs          size of table
I_PHASE         equ     3               3 chars enciphered per inc

A               equ     65              ASCII 'A'
Z1              equ     91              ASCII 'Z' + 1
a               equ     97              ASCII 'a'
z1              equ     123             ASCII 'z' + 1
e               equ     101             ASCII 'e'
d               equ     100             ASCII 'd'

XDC             equ     77              flag value for deciphering
ENC             equ     88              flag value for enciphering
CR              equ     13              carriage return
LF              equ     10              line feed
TAB             equ     9               tab
BS              equ     8               backspace
SPACE           equ     32              space
```

ST USER

by Arthur Leyenberger

It feels great to be back in the saddle again. **ST-Log** is back on schedule, with new backing, a more professional look and all systems go. Renewing my contacts and friends on Delphi (my user ID is ARTL) and on CompuServ (My ID is 71266,46) has been fun, and I look forward to spending more time talking with ST users about the topics that concern all of us.

Diamond in the rough

When I first heard that Abacus Software was preparing a word processor for the Atari ST computer, I thought to myself, *Oh, no, not another one.* All too often a company that does one thing well attempts to go back to the well and compete with still *another* word processor, terminal program or whatever. I'm happy to report that Abacus not only excels in the technical ST book category—they are clearly the best and the most prolific publisher—but that they also have a very good word processor. My fears, it seems, were unfounded.

Like several other Abacus products, **Text Pro** was originally written by Data Becker, a German software publishing company. The three program authors are all professional writers, we are told, so the program reflects what they wanted in a word processor. Although the apparent translation of the documentation is adequate, I would have expected a little more substance, especially from professional writers.

Text Pro is a fully GEM-based word processor with drop-down menus located across the top of the screen, windows with vertical and horizontal scroll bars, and access to any Desktop accessories you may have already loaded when you booted up the ST. However, once you learn the program, you can use keyboard commands instead of "mousing" around. Text is entered in a continuous stream, with formatting commands embedded within the text. Although it is not a what-you-see-is-what-you-get (WYSIWYG) program, the appearance of your document on the screen is similar to the final paper output.

An essential part of Text Pro is what's called the Format Template. This is where you can change the appearance of a document to be printed. A Format Template is saved with the text file and determines the number of lines per page, line spacing, margins, header and footer spacing, column width and number of columns (for multiple-column printing) per page. Many format templates can be saved on your disk, since the "look" of a letter differs from that of a report or term paper.

Piracy is not new to the Atari or any user community.

Text Pro is capable of a few more output tricks. Text can be printed vertically (normal) or horizontally to either the printer or a file. In addition, a file can be saved as a "text design" file. Here, the output is saved as a bit-mapped representation of the screen for further editing by **Text Designer**, another Abacus product. By use of a "placeholder" command, a second file may be merged with your text using the output program.

There is a special feature for C programmers too. In this mode braces are automatically indented. This feature and the 30 programmable function keys make programming in C much easier.

The first Text Pro feature that grabbed my attention was the ability to print in multiple columns. All of the first-generation ST word processors lacked this feature (except for **ST-Writer**, which was somewhat bugridden). Now, newsletter editors in particular, or anyone else on a tight budget for that matter, can take advantage of formatting your text in however many columns you require. The procedure is straightforward and the results are impressive.

Text Pro provides the usual block, search and replace, word wrap, justification and other word-processing functions you would expect with this type of program. In addition, certain text attributes, such as normal or bold print, are displayed on the screen. An upper-lower case toggle lets you switch text from one to the other, and the system date and time can be inserted anywhere in the text.

Text Pro comes out of the box ready to work with an Epson compatible printer. If you have some other type of printer you'll have to customize the program for that printer or embed printer codes within the text. The first option entails creating a printer driver by modifying the default Epson driver supplied with the program. There are two ways to approach this. In one, you copy the original driver file to another file name (as backup) and modify the original, since Text Pro is looking for that (original) specific filename. Alternatively, the program has an output feature that lets you choose from a number of printer drivers already created. In either case, you have to create a new driver for your non-Epson printer—although it is not difficult.

A pair of additional utility programs are supplied with Text Pro in order to make it more useful. SPLIT is a program designed to split a very large file in half, and split those halved files in half, un-

til you have files of a manageable size for the amount of memory remaining in your ST. The other program, CONV, lets you convert files from other word processors, such as **1st Word,** into Text Pro format. Page breaks and text attributes are acknowledged and carriage returns at the end of lines are changed to floating text. Both of these programs are run outside of Text Pro.

Drawbacks

There are several missing features in this word processor. First off, there is no Undo function. Once you erase something, it's history. There is no way to save a file under a different name while editing a document. Often, you want to take a file, make a few changes here and there and save it under a different name. Text Pro forces you to copy the file first, from the GEM Desktop, then edit the newly created file. No major problem, merely a minor hassle.

There is no way to specify a last page to print when printing to the screen or the printer since the program prints from a starting page to the end. Further frustration occurs because there is no graceful way to stop the printout. There is no provision for a clipboard or second window to aid text editing and copying. And there is no on-line help.

Bottom Line

Despite the few flaws (really only missing features rather than bugs), Text Pro is a useful, second-generation word processor for the Atari ST. Its ability to print two-line headers and footers, display output on the screen, print multiple-column text and perform logical hyphenation makes the program one of the best currently available for the ST. In addition, such features as sorting, indexing and creating a table of contents make Text Pro worthy of your serious attention when choosing a word processor. Also, its list price of under $50 won't require a second mortgage on the condo.

More WordPerfect and Piracy

Recently, CompuServe was abuzz with the potential withdrawal of a major ST software publisher. WordPerfect Corp. had contacted SYSOP Ron Luks and told him that because of software piracy, they were planning to remove the word processor **WordPerfect** from the ST market. Never in my five years of using CompuServe have I seen such a quantity of responses to Ron's initial message. It all culminated in a CO (conference) with several WordPerfect representatives and about 50 CompuServe users to discuss the issues.

Piracy is not new to the Atari or any user community. It has the potential for drying up the supply of software and hurting those legitimate users who paid for their hardware *and* the software they used. In fact, some people believe that the lack of 8-bit Atari software is

We had better put a stop to piracy now.

due directly to the amount of piracy that has occurred for years in this market.

Anyway, it seemed that WP Corp. had discovered several pirate bulletin boards that were making their WordPerfect program freely available to anyone that called. This had (understandably) upset them, and they decided that the ST software market was one that they no longer wanted to participate in. As it turned out, the decision to leave the ST market had not been made at that point, and the positive responses from CompuServe members helped persuade WordPerfect Corp. that the majority of ST users are not thieves.

However, a number of interesting and significant points were raised in the message base and also when I spoke to Todd Ashman (Director of Atari marketing for WP) and Jeff Wilson (manager of Atari Development for WP). One point that was repeatedly made by the WP folks is that the initial release of WordPerfect had a number of bugs in it. They freely admit that it was released prematurely and was not thoroughly tested. Because of the relatively high price ($395 list, average street price about $250) and the type of excellent (my words) support given to their products, WP was able to release two upgrades to all registered users at no charge to them. They have also provided toll-free support lines to help users install and use the product.

Although there were no major bugs in the software, there were quite a few little things that would accumulate and ultimately crash the program. The printer support section of the program has been rewritten to match the function of the PC version and have less variability in operation. Some of the ST operating-system calls had to be redone and replaced in order to prevent the program from hanging, due to bugs in the Atari operating-system software.

Another version of WordPerfect, which was due to be released during the last week of March, is said to be virtually problem-free. All registered users will receive this upgrade automatically, at no cost to them. The latest version of the program will also incorporate many suggestions from users to improve its operation.

The piracy issue basically comes down to the fact that the PC market is 50 times bigger than the ST market. In this market, even a large percentage of piracy does not prevent the company from making a profit. Unfortunately, the smaller ST market feels the effects of piracy first. Piracy has the effect of causing a proportionately larger percentage of lost sales.

WordPerfect is currently the only major-league software publisher that has a product in the Atari ST marketplace. That's major league, as in MS-DOS software publisher selling a product consistently in the top ten list. If we as ST users wish to continue to have companies like WordPerfect, with their high level of support, quality product (now), and open-minded attitude support our computers in the future, we had better put a stop to piracy *now*.

It's one thing to be against stealing software yourself and not do it. But we all need to be vocal about it and spread the word. Atari users are not necessarily any worse than other computer users when it comes to piracy. But the bottom line is that stealing software is not only wrong, it can have harmful effects on the longevity and health of your computer.

Piracy. Just say *no.*

MAGNAVOX

8CM873
MULTI-SYNC

- Multimode text/graphics design permits operationel compatibility with all currant computer standards
- Rastar scan operation in 3 frequancy bands from 15kHz to 34kHz
- Accepts RGB vidao in digital and analog form
- Accepts line level audio input
- Image resolution of up to 926 dots (horizontel) and 580 linas (vertical)
- Grean text mode display switch
- Convaniant tilt/awivel base availebla

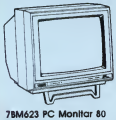
2 year warranty

$475

EGA Card – $125

8CM515
New
$269.95

Cable 515 Magnavox to Amiga Computer – $13.95



$79.95

7BM623 PC Monitar 80
• 12" Monochrome TTL input manitor • Lang persistance LA Amber phosphar • Dark glass CRT • Nan-glare treated faceplate • 22MHz amplifier bandwidth • Displays 2000 characters, 80×25 • 1000 lines of resalution

Modems

2400 Baud External	139.95
2400 Baud Internal	139.95
1200 Baud External	79.95
1200 Baud Internal	64.95

100% Hayes Compatibility
Hayes is a Registered Trademark of Hayes

- **Full manufacturer warranty.**
- **Personal check 3 weeks clearance.**
- **Return authorization required.**
- **Prices show 3% cash discount.**
- **Compatibility not guaranteed.**
- **COD accepted.**

APO, FPO, International: add $5 plus 3% priority.
No sales tax outside N.Y.
Prices/availability subject to change.

MURATA FAX MACHINES NOW IN STOCK.
CALL FOR PRICING.

star micronics

BEST PRICE CALL

NX1000 Rainbow Color Printer – Call

New

NX1000
NLQ – 36 CPS
DRAFT – 144 CPS

NICKEL CITY ELECTRONICS

Orders
1-800-634-1870

Tech Support
1-716-684-7350

P.O. Box 1025, Buffalo, NY 14225

SUPER SPECIAL OF THE MONTH

2400 Internal Modem
100% Hayes Compatibility

$119

Panasonic
Office Automation

KXP 1080 II
$174.95

Economy and quality work togethar in the KX-P1080 dot matrix printer. Economy, because the KX-P1080 is priced to suit even a modest budget. Quality, because it's from Panasonic. With a 2-year-limited warranty to back it up. So come and see the KX-P1080 – it's a great value worth a closer look.

- Prints 100 cps in draft mode
- Near Letter quality printing
- Operator accessible print mode switch
- Dot addressable graphics
- 2-year limited warranty
- Word processing functions (justification, centering, margin alignment)

KXP1091II – $194.95

Call for pricing on other models

Accessories

A/B DB9 Switch Box	32.95
ABCD DB9 Switch Box	49.95
A/B Telephone Switch Box RJ11	34.95
Serial Cross-Over Switch Box	39.95
Parallel Cross-Over Switch Box	39.95
M-M DB9 Gender Changer	5.95
F-F DB9 Gender Changer	5.95
A/B Parallel Switch Box	22.95
A/B Serial Switch Box	22.95
A/B/C/D Parallel Switch Box	39.95
A/B/C/D Serial Switch Box	39.95
6 Outlet AC Surge Pr	9.95
B-109 Parellel Card	29.95
B-106 Serial Card	26.95
Male/Male Gender Chg	5.95
Feml/Feml Gender Chg	5.95
Winner 200	11.99
Winner 104	4.99
Winner 770 Com/Atari	11.95
Winner 909 Apple/IBM	23.95
Winner 220 Joystick	13.95
500XJ Atari/Comm	14.95
Competition Pro 5000	14.95
Kraft Apple/IBM KC 3	24.95
Mazemaster Atari/C64	14.95
#31 Apple Mouse	32.95
Starmaster Joystick	5.49
Joystick Ast Ada. Kit	27.95
Kraft Ace Joystick	5.49
Kraft Premium 2 IBM	26.95
Apple Mach 3 Stick	12.95
Apple Kraft Stick	8.95
Mouse Joystick	12.95
Competition Pro	15.95
Suncom Stik Stiks	5.99
Suncom Econo Stik	4.99
Suncom Tac 2	9.99
Suncom Tac 3	11.99
Suncom Tac 5	13.95
Suncom Starfighter	17.85
Icon Controller	15.99
Tac 1	24.95
Starfighter IBM	17.85
Starfighter C64/Atar	8.95
Terminetor Joystick	19.95

Cables

IBM Parallel Cable 6'	5.99
IBM Parallel Cable 10'	7.99
M-M RS232 Cable 6'	7.95
M-M RS232 Cable 10'	8.49
M-F RS232 Cable 6'	7.95
M-F RS232 Cable 10'	8.49
IBM Modam Cable 6'	5.99
IBM Modem Cable 10'	7.99
IBM at Modem Cable	5.99
M-M Centronics 6'	11.95
M-F Centronics	11.95
M-M Centronics 10'	12.95
IBM Keyboard Extension	4.99
Monitor Extension-DB9 M-F	11.95
Monitor Cable-DB9 M-M	11.95
IBM Color RGB Cable-DB9 Cable to 8 Pin Din	12.95
Mac+ to Imegewriter	9.99
Mac+ to Imagewriter II	9.99
Mac+ to Hayse	9.99

CIRCLE #106 ON READER SERVICE CARD.

Disk Drives that are Setting new Standards for ST Enhancements

Performance *Styling* *Reliability*

- Introducing the 5¼" Disk Drive for the Atari ST Computer Systems.
- The smoked glass dust cover protects the GT's front loading front end. Preventing both dust and misguided fingers.
- The GT's Accutouch buttons and LED lights provide absolute feedback. Giving you positive assurance that data integrity has been preserved.
- When the GT's Protect LED is lit, your drive will not be able to write on diskettes. Electronically or mechanically. Protecting, you and your program from "Driver Error"
- The GT-1000 comes with an external power supply and is available in either 40 Track, 360 k. capacity or 80 Track 720 k. capacity. (Optional)

The all new 1988 GT disk drives.

- The GTS-100 is sleek and perfectly proportioned. Eye catching. Distinctively European Gran Styled.
- The front panel display, track readouts. Optimizing your concentration. Eliminating doubt and second guessing.
- The busy LED notifies when your ST is using the GTS-100. And it also doubles as a warning not to remove your diskette or power-off your system while it is lit.
- The 3½" Diskette transport has been equipped with the *GTS Cruise Control* head positioning mechanism – the lowest friction, highest accuracy positioner available.
- The GTS is dual sided, double density floppy disk drive with a 720 k formatted storage capacity.
- Comfortable. Impressive. Friendly. Three excellent reasons to go see and test drive the new GTS-100 at your ST dealer today.

Future
SYSTEMS, INC.

SERVICE *VALUE* *QUALITY*

ATARI ST is a trademark of ATARI Corp.

For dealer information call: (818) 407-1647

GTS-100 & GT-1000 is a trademark of & is manufactured by F.S.I., 21634 Lassen, Chatsworth, CA 91311

CIRCLE #107 ON READER SERVICE CARD.